ВВЕДЕНИЕ В JAVASCRIPT

Власов Виталий
2019
ФБИТ, ИТМО

КАК ДОБАВИТЬ JS КОД

- Внутри страницы в тэге <script>
- Внутри тэга в аттрибутах
- В отдельном файле .js.

```
<script>
    alert ("Привет мир!");
</script>
```

```

<a href="/delete"
    onclick="return confirm('Вы уверены?');"> Удалить </a>

```

```

<script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>

```

ЗАПОМНИТЕ

- В устаревших руководствах (до HTML 5) можно встретить следующие элементы: атрибут <*script type=“text/javascript*">. В современной разработке атрибут type необязателен. Атрибут <script *language*=…> предназначен для указания языка, на котором написан скрипт, так как по умолчанию это JavaScript то и этот атрибут ставить необязательно.

- Браузер, когда видит <script> - начинает отображать страницу, показывает часть документа до script. Встретив тег <script>, переключается в JavaScript-режим и не показывает, а **исполняет** его содержимое. Закончив выполнение, возвращается обратно в HTML-режим и только тогда отображает оставшуюся часть документа.

- Для асинхронной загрузки внешних скриптов можно использовать async/defer. Изучить самостоятельно.

ЗАПОМНИТЕ

- В HTML пишут только самые простые скрипты, а сложные выносят в отдельный файл. Браузер скачает этот файл только первый раз и в дальнейшем, при правильной настройке сервера, будет брать из своего кеша. Благодаря этому один и тот же большой скрипт, содержащий, к примеру, библиотеку функций, может использоваться на разных страницах без полной перезагрузки с сервера.

- Если указан атрибут **src**, то содержимое тега игнорируется. В одном теге <script> нельзя одновременно подключить внешний скрипт и написать код.

```
4  <script src="script.js"></script>
5  <script>
6      alert ("Привет мир!");
7  </script>
```

ПЕРЕМЕННЫЕ

- Правило 1 Никакого транслита. Только английский.
- Правило 2 Использовать короткие имена только для переменных «местного значения».
- Правило 3 Переменные из нескольких слов пишутся вместеВотТак (myBestResult)
- Правило 4, главное. Имя переменной должно максимально чётко соответствовать хранимым в ней данным.

```
var user = "Vitaly", year = 2019;
```

ТИПЫ ДАННЫХ

- **Number** - единый тип число используется как для целых, так и для дробных чисел. Существуют специальные числовые значения Infinity (бесконечность) и NaN (ошибка вычислений)
- **String** В JavaScript одинарные и двойные кавычки равноправны. Тип символ не существует, есть только строка.
- **Boolean**
- **Null** В JavaScript null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках. Это просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».
- **Undefined** Оно имеет смысл «значение не присвоено». Если переменная объявлена, но в неё ничего не записано, то её значение как раз и есть undefined
- **Object** Он используется для коллекций данных и для объявления более сложных сущностей. Объявляются объекты при помощи фигурных скобок {…}
- Оператор **typeof** возвращает тип аргумента. Синтаксис typeof x или typeof(x). Работают они одинаково., но первый синтаксис короче. Результатом typeof является строка, содержащая тип.

```
var user = { name: "Vitaly" };
```

МОДАЛЬНЫЕ ОКНА

- **alert**(сообщение) Выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК»
- **prompt**(title, default); Она выводит модальное окно с заголовком title, полем для ввода текста, заполненным строкой по умолчанию default и кнопками OK/CANCEL
- **confirm**(question); Выводит сообщение и ждёт, пока пользователь нажмёт «ОК» или «CANCEL» и возвращает true/false

```
var years = prompt('Сколько вам лет?', 100);
alert('Вам ' + years + ' лет!');


var isAdmin = confirm("Вам"+ years + ' лет?');
alert(isAdmin);
```

УСЛОВНЫЕ ОПЕРАТОРЫ: IF, '?'

- Рекомендуется всегда использовать фигурные скобки
- Условные операторы: if, '?' Оператор if (...) вычисляет и преобразует выражение в скобках к логическому типу. В логическом контексте: Число 0, пустая строка "", null и undefined, а также NaN являются false, Остальные значения – true. Для этого используется блок else if …
- «Тернарный оператор» Вопросительный знак – единственный оператор, у которого есть аж три аргумента, в то время как у обычных операторов их один-два. Поэтому его называют «тернарный оператор».

```
if (year == 2019) {

    alert (text);
    alert (text1);
    alert (text2);

} else if (year != 2018 ) {

    alert (text);

} else {

    alert (text);
}
```

```
1 if (0) { // 0 преобразуется к false
2   ...
3 }
```

...А такое – выполнится всегда:

```
1 if (1) { // 1 преобразуется к true
2   ...
3 }
```

```
условие ? значение1 : значение2
```

```
access = (age > 21) ? true : false;
```

- «Тернарный оператор» Последовательность операторов '?' позволяет вернуть значение в зависимости не от одного условия, а от нескольких.

```
var age = prompt('возраст?', 18);

var message = (age < 3) ? 'Здравствуй, малыш!' :
  (age < 18) ? 'Привет!' :
  (age < 100) ? 'Здравствуйте!' :
  'Какой необычный возраст!';

alert( message );
```

```
if (age < 3) {
  message = 'Здравствуй, малыш!';
} else if (age < 18) {
  message = 'Привет!';
} else if (age < 100) {
  message = 'Здравствуйте!';
} else {
  message = 'Какой необычный возраст!';
}
```

ЦИКЛЫ

- Повторение цикла по-научному называется «итерация».
- Существует также специальная конструкция for..in для перебора свойств объекта.
- Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент. Эту возможность обеспечивает директива break
- Директива continue прекращает выполнение текущей итерации цикла. Она прерывает не весь цикл, а только текущее выполнение его тела, как будто оно закончилось. Её используют, если понятно, что на текущем повторе цикла делать больше нечего.
- Синтаксические конструкции, которые не возвращают значений, нельзя использовать в условии.

```
while (условие) {
  // код, тело цикла
}
```

```
do {
  // тело цикла
} while (условие);
```

```
for (начало; условие; шаг) {
  // ... тело цикла ...
}
```

```
for (var key in menu) {
    // этот код будет вызван для каждого свойства объекта
    // ..и выведет имя свойства и его значение

    alert("Ключ: " + key + " значение:" + menu[key]);
}
```

ФУНКЦИИ

- Вначале идет ключевое слово function, после него имя функции, затем список параметров в скобках (в примере выше он пустой) и тело функции – код, который выполняется при её вызове.

```
function nameOfFunction(param1, param2) {
    тело функции
    return;
}
```

РАБОТА СО СТРОКАМИ

```
alert( "Привет, мир!".length ); // 12
alert( "Привет, мир!".toUpperCase() ); // "ПРИВЕТ, МИР!"
alert( "Привет, мир!".charAt(0) ); // "П"

var str = "Widget with id";
alert( str.indexOf("Widget") ); // 0, т.к. "Widget" найден прямо в начале str
alert( str.indexOf("id") ); // 1, т.к. "id" найден, начиная с позиции
alert( str.indexOf("Lalala") ); // -1, подстрока не найдена

var str = "stringify";
alert(str.substring(0,1)); // "s", символы с позиции 0 по 1 не включая 1.
```

ОБЪЕКТЫ

- Объекты - это ассоциативные массивы (хранит пары «ключ-значение»).
- Создание объекта: obj = new Object();
- Создание объекта: obj = {}; // пустые фигурные скобки

```
var person = {};
// при присвоении свойства в объекте автоматически создаётся "ящик"
// создадим свойство с именем "name" и в него записывается содержимое 'Вася'
person.name = 'Вася';
// можно обратиться и так
person['name'] = 'Вася';
person.age = 25;
alert(person.name + ': ' + person.age); // вывести значения
delete person.name;
// удалить "ящик" с именем "name" вместе со значением в нём
```

ПРОВЕРКА СУЩЕСТВОВАНИЯ СВОЙСТВА С ОПРЕДЕЛЕННЫМ КЛЮЧОМ

```
var person = { };

if ("age" in person) {
    alert("Свойство age существует!");
}
```

```
var person = {
    age: 25,
    name: "Vitaly",
    'sex': "male",
    'full address': "my home address",
    contact: {
        email: "male@male.com",
        phone: "112",
      }
};

var key = 'age';

alert(person[key]);  // выведет person['age']

alert(person.contact.email);
```

МАССИВЫ

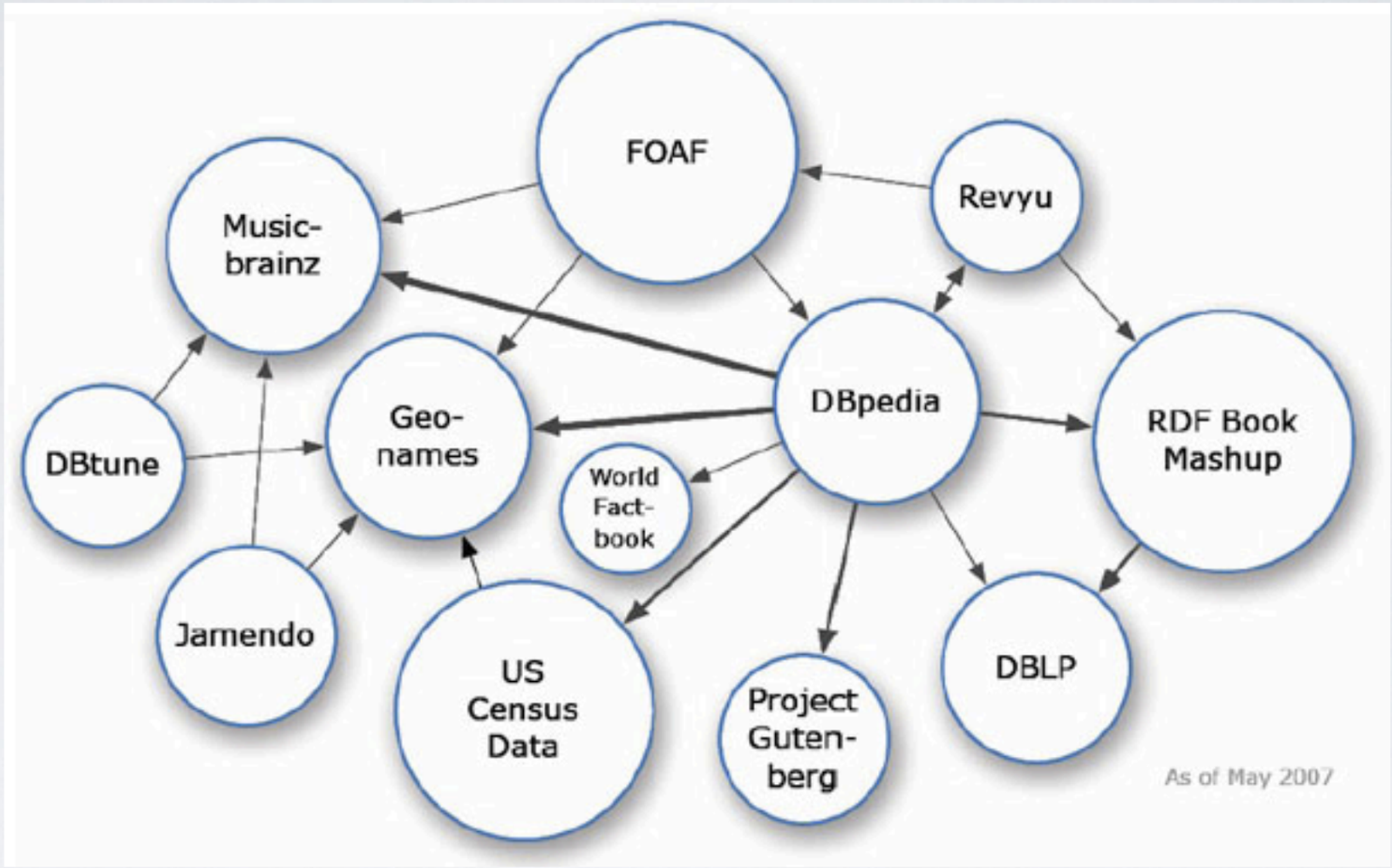
```
var arr = [];

var fruits = ["Яблоко", "Апельсин", "Слива"];

alert(fruits[0]);

var arr = [ 1, 'Имя', { name: 'Петя' }, true ];

alert( arr[2].name );
```

- **pop()** - удаляет последний элемент массива
- **push (element)** - добавляет элемент массива в конец
- **shift()** - удаляет первый элемент массива
- **unshift(element)** - добавляет элемент массива в начало
- Допустимо: **fruits.push("Апельсин", "Персик");**
  fruits.unshift("Ананас", "Лимон");

ФУНКЦИЯ — ЭТО ЗНАЧЕНИЕ И ОБЪЕКТ

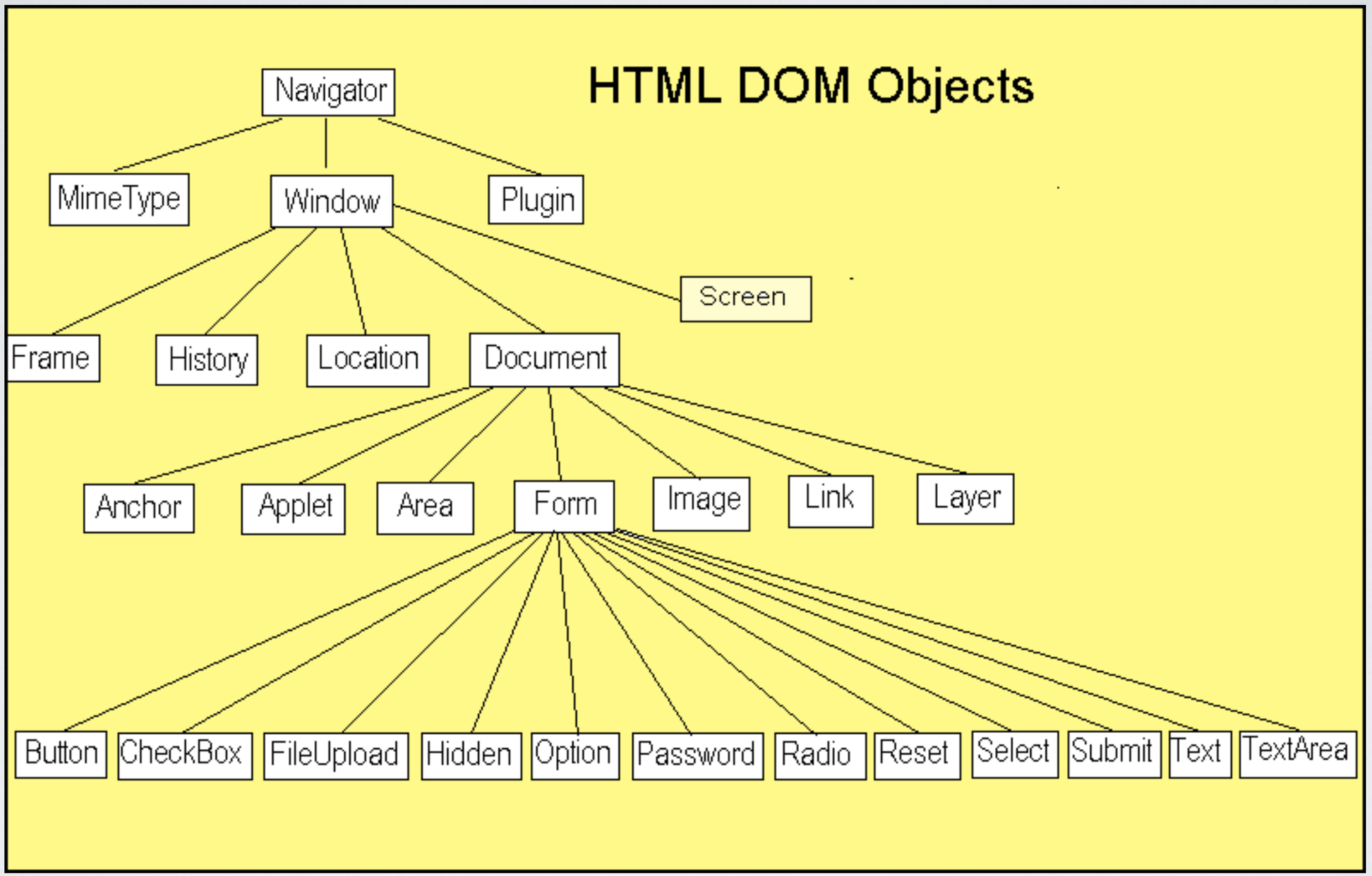
```
function sayHi() {
  alert('Привет');
}
alert(sayHi); // выведет код функции
```

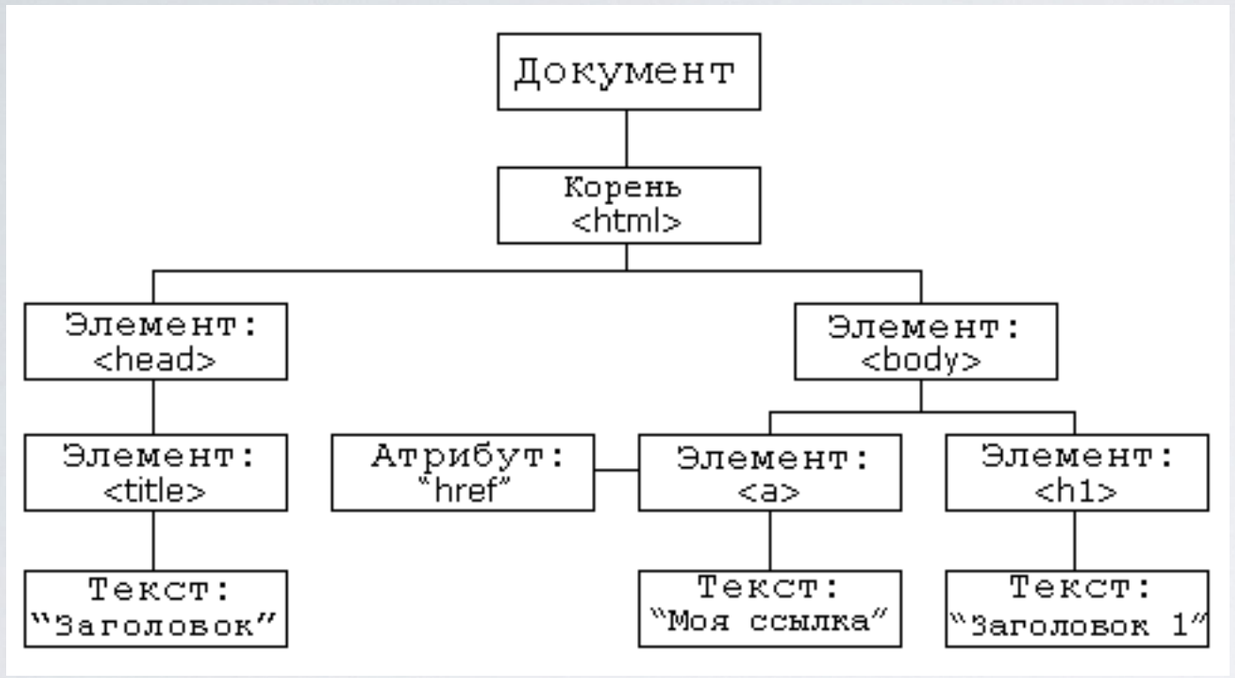
```
function sayHi() { }
sayHi.test = 5;
alert(sayHi.test); // выведет 5

var func = sayHi; // функцию можно копировать
```

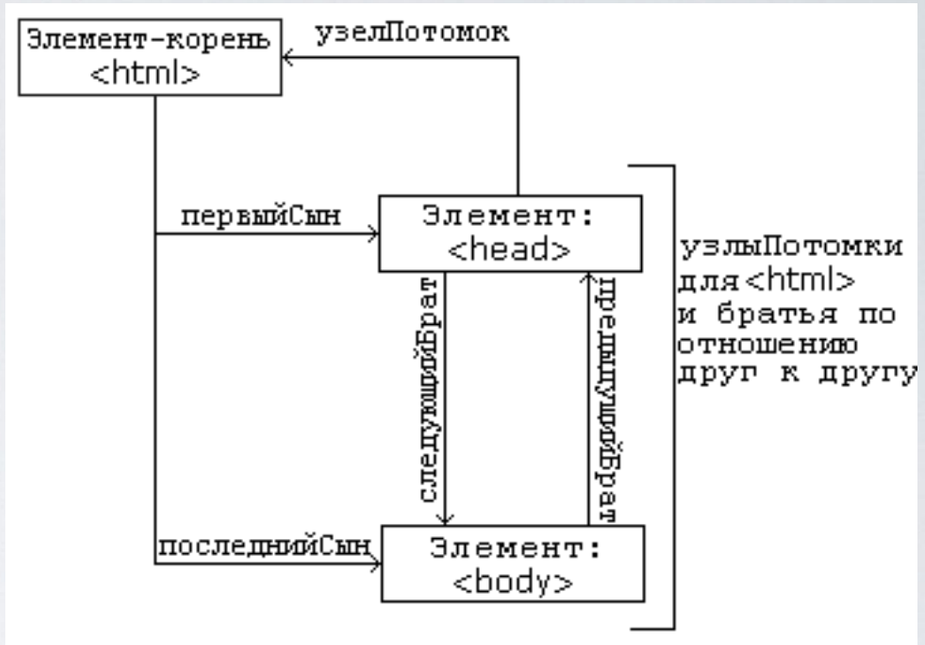


DOM

Document Object Model


HTML DOM Objects
Navigator
MimeType
Window
Plugin
Screen
Frame
History
Location
Document
Anchor
Applet
Area
Form
Image
Link
Layer
Button
CheckBox
FileUpload
Hidden
Option
Password
Radio
Reset
Select
Submit
Text
TextArea





- В дереве узлов самый верхний элемент называется корнем
- Каждый узел, кроме корня, имеет в точности одного предка
- Узел может иметь сколько угодно потомков
- Лист - это узел, у которого нет ни одного потомка
- Братьями являются узлы с одним и тем же предком

HTML DOM СВОЙСТВА

- x.innerHTML - текстовое значение x
- x.nodeName - название (имя) узла x
- x.nodeValue - значение x
- x.parentNode - родительский узел x
- x.childNodes - дочерние узлы x
- x.attributes - атрибутивные узлы x

HTML DOM МЕТОДЫ

- x.getElementById(id) - получить элемент с указанным id
- x.getElementsByTagName(имя) - получить все элементы с указанным названием (именем) тега
- x.appendChild(узел) - вставить дочерний узел в x
- x.removeChild(узел) - удалить дочерний узел из x

```
<html>
<body>

<p id="intro">Привет Мир!</p>

<script type="text/javascript">
    txt=document.getElementById("intro").innerHTML;
    document.write("<p>Текст из параграфа intro: " + txt + "</p>");
</script>

</body>
</html>
```

CHILDNODES И NODEVALUE

```
<html>
<body>

<p id="intro">Примет Мир!</p>

<script type="text/javascript">
    txt=document.getElementById("intro").childNodes[0].nodeValue;
    document.write("<p>Текст из параграфа intro: " + txt + "</p>");
</script>

</body>
</html>
```

DOM ДЛИНА СПИСКА УЗЛОВ

```
x=document.getElementsByTagName("p");
for (i=0;i<x.length;i++)
{
document.write(x[i].innerHTML);
document.write("<br />");
}
```

ИЗМЕНИТЬ HTML ЭЛЕМЕНТ

```
<html>
<body>

<script type="text/javascript">
document.body.bgColor="lavender";
</script>

</body>
</html>
```

INNERHTML

```
<html>
<body>

<p id="p1">Привет Мир!</p>

<script type="text/javascript">
document.getElementById("p1").innerHTML="Новый текст!";
</script>

</body>
</html>
```

СОБЫТИЯ

```
<html>
<head>
<script type="text/javascript">
function ChangeText()
    {
        document.getElementById("p1").innerHTML="Новый текст!";
    }
</script>
</head>
<body>
<p id="p1">Привет мир!</p>
<input type="button" onclick="ChangeText()" value="Изменить
текст" />
</body>
</html>
```

```
<html>
<head>
<script type="text/javascript">
function ChangeBackground()
    {
        document.body.style.backgroundColor="lavender";
        document.getElementById("p1").style.color="blue";
        document.getElementById("p1").style.fontFamily="Arial";
    }
</script>
</head>

<body>
<input type="button" onclick="ChangeBackground()"
value="Изменить цвет фона" />
</body>
</html>
```

onClick, onFocus, onLoad, onBlue, etc

ЗАДАНИЯ

- Показать текст в браузер с помощью document.write()
- Показать отформатированный текст в браузер с помощью document.write()
- Показать количество анкоров в документе
- Показать содержимое (innerHTML) первого анкора в странице
- Показать количество форм в документе
- Показать название (имя) первой формы в документе
- Показать количество изображений в документе
- Показать id первого изображения в документе
- Показать количество ссылок в документе
- Показать id первой ссылки в документе
- Показать имя домена сервера, который загрузил документ
- Показать URL документа, который загрузил текущий документ
- Показать заголовок документа
- Показать полный адрес URL документа

ЗАДАНИЯ

- **Объект Кнопка (button)**
- Сделать кнопку в состояние disabled при ее нажатии
- Показать имя кнопки
- Показать тип кнопки
- Показать текст, отображаемый на кнопке
- Показать id формы, которой принадлежит кнопка

- **Объект Форма (form)**
- Показать значение каждого элемента на форме
- Показать значение атрибута action формы
- Показать значение атрибута enctype формы
- Показать количество элементов на форме
- Показать метод отправки данных с формы
- Показать имя (название) формы
- Сбросить форму
- Отправить данные с формы

ЗАДАНИЯ

- **Объект Изображение (image)**
- Выравнять изображения
- Показать альтернативный текст для изображения
- Добавить границу к изображению
- Изменить высоту и ширину изображения
- Показать название изображения
- Изменить источник изображения
- **Объект Событие (event)**
- Какая кнопка мыши была нажата?
- Определить код нажатой клавиши
- Определить координаты курсора
- Определить координаты курсора относительно экрана
- На каком элементе был клик?
- Определить тип произошедшего события

ЗАДАНИЯ

- **Объекты Таблица (table), Заголовок Таблицы (header), Строка Таблицы (row), Данные Таблицы (cell)**
- Изменить ширину границы таблицы
- Изменить cellPadding и cellSpacing таблицы
- Указать рамку таблицы
- Создать заголовок для таблицы
- Удалить строки в таблице
- Добавить строки в таблицу
- Добавить ячейки в строку таблицы
- Выравнивание содержимого ячейки в строке таблицы
- Вертикальное выравнивание содержимого ячейки в строке таблицы
- Выравнивание содержимого ячейки в одиночной ячейке
- Вертикальное выравнивание содержимого ячейки в одиночной ячейке
- Изменить содержимое ячейки таблицы
- Изменить colspan строки таблицы