информационные технологии
КЕЙСИСТЕМС

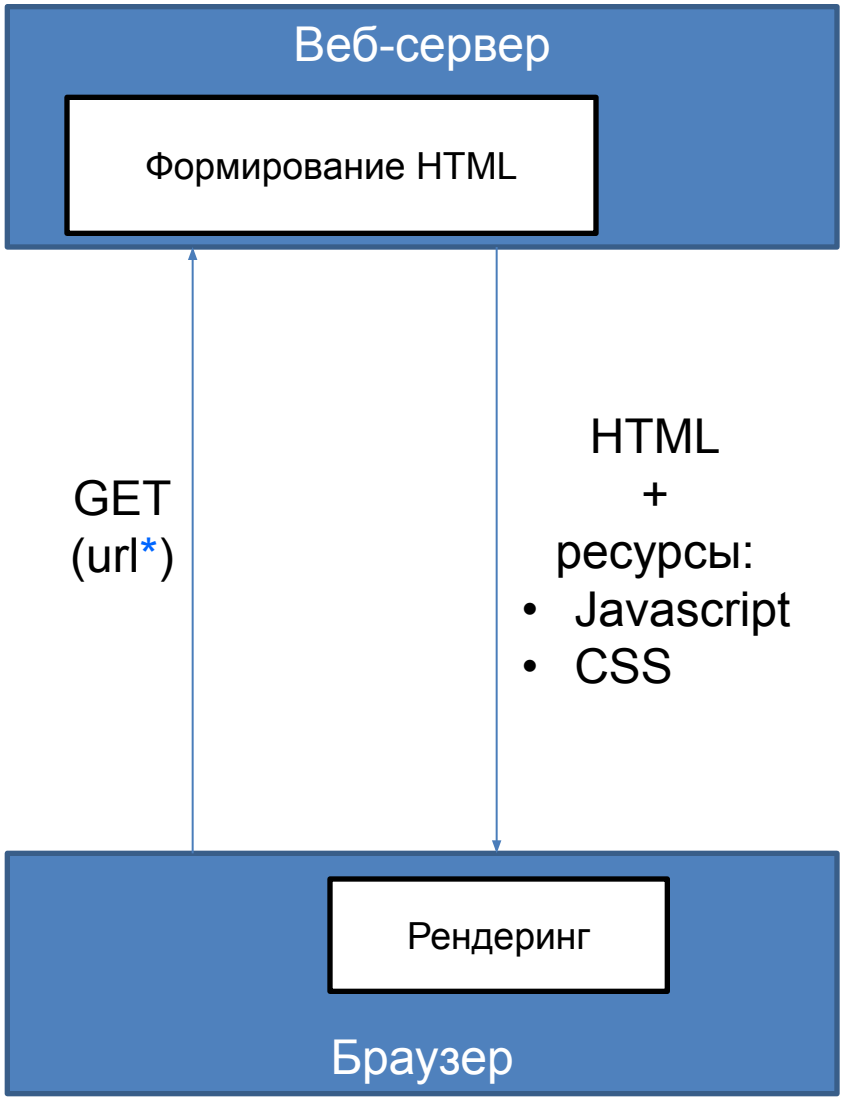
Веб-программирование

2. Принципы функционирования фронтенд, рендеринг HTML, DOM.

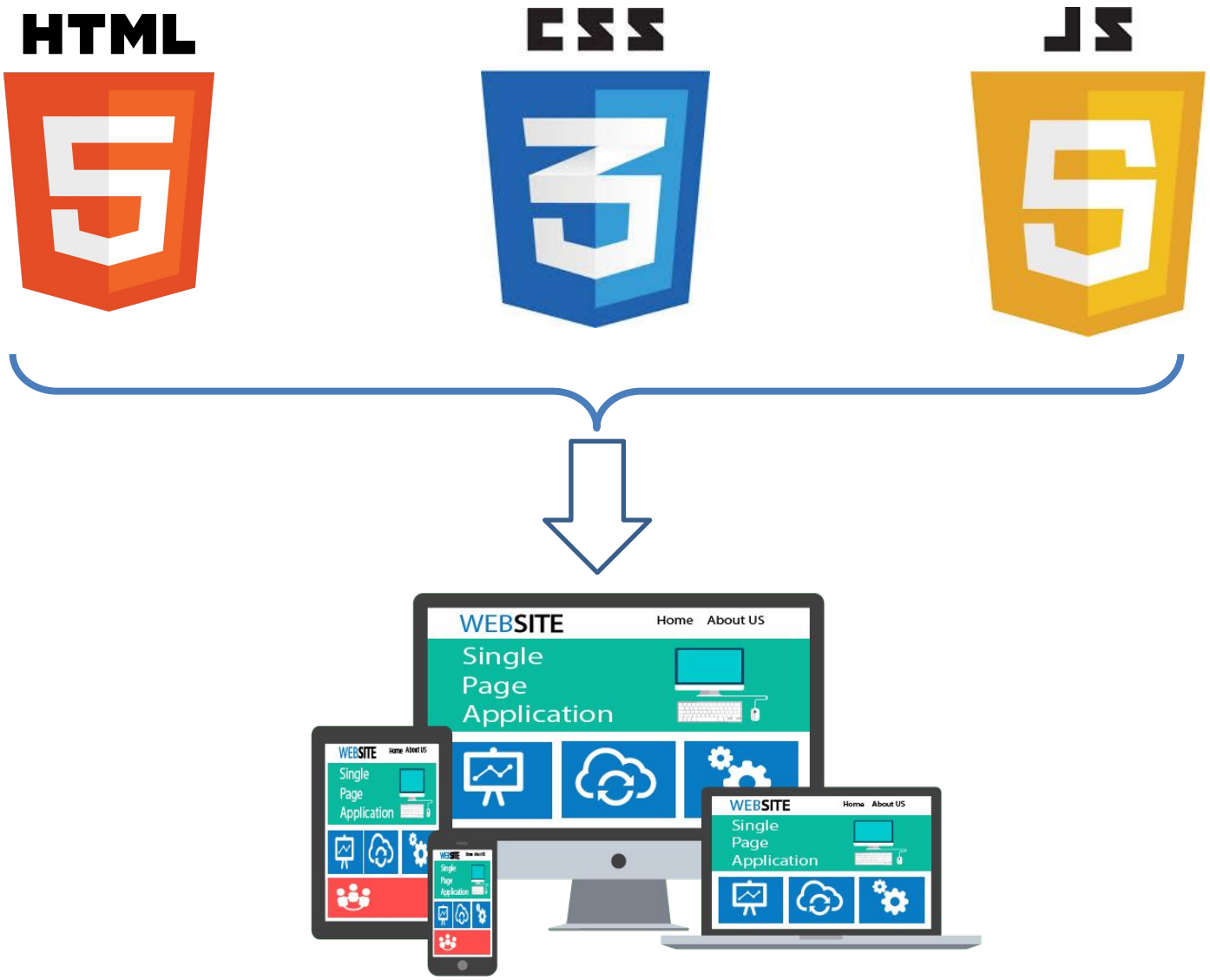
Докладчик:

Цветков Игорь Владимирович

ОТКРЫТИЕ ВЕБ-СТРАНИЦЫ



БАЗОВЫЕ ТЕХНОЛОГИИ ФРОНТЕНД (БРАУЗЕР)



HTML

```
<!doctype html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">

    <style type="text/css">
        #hello {
            font-size: 16px;
        }
    </style>
</head>

<body>
    <h1 id="hello">Hello world</h1>
    <div id="root"/>
</body>
</html>
```

HTML ЭЛЕМЕНТЫ (ТЕГИ)

**Тег ** универсальный блочный элемент.

Тег <h1> заголовок первого уровня.

Тег <p> определяет текстовый абзац.

Тег **<button>** создает на веб-странице кнопку

Атрибуты тегов

Чтобы расширить возможности отдельных тегов и более гибко управлять содержимым контейнеров применяются атрибуты тегов.

- Атрибут **id** применим к любому html-тегу и позволяет задать уникальный идентификатор в пределах страницы и может использоваться для его поиска:

```
<h1 id="hello">Hello world</h1>
```

- Атрибут **class** означает, что отображение элемента определяется CSS-классом

```
<div id="root" class="green-background"/>
```

- Еще один важнейший тип атрибутов – **обработчики событий**

```
<button onclick='findElement();'>Вывести значение</button>
```

CSS (КАСКАДНЫЕ ТАБЛИЦЫ СТИЛЕЙ)

CSS — формальный язык описания внешнего вида документа, т.е. для задания цветов, шрифтов, расположения блоков и других аспектов представления внешнего вида этих веб-страниц.
Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (с помощью HTML) от описания внешнего вида этой веб-страницы (с помощью CSS).

Каждое правило CSS имеет две основные части:

- *селектор* (до знака “{“)
- *блок объявлений (внутри { })*

Основное различие между классами элементов и идентификаторами элементов в том, что идентификатор предназначен для одного элемента, тогда как класс обычно присваивают сразу нескольким.

Селектор идентификаторов:

```
#hello {
    font-size: 16px;
}
```

Селектор классов:

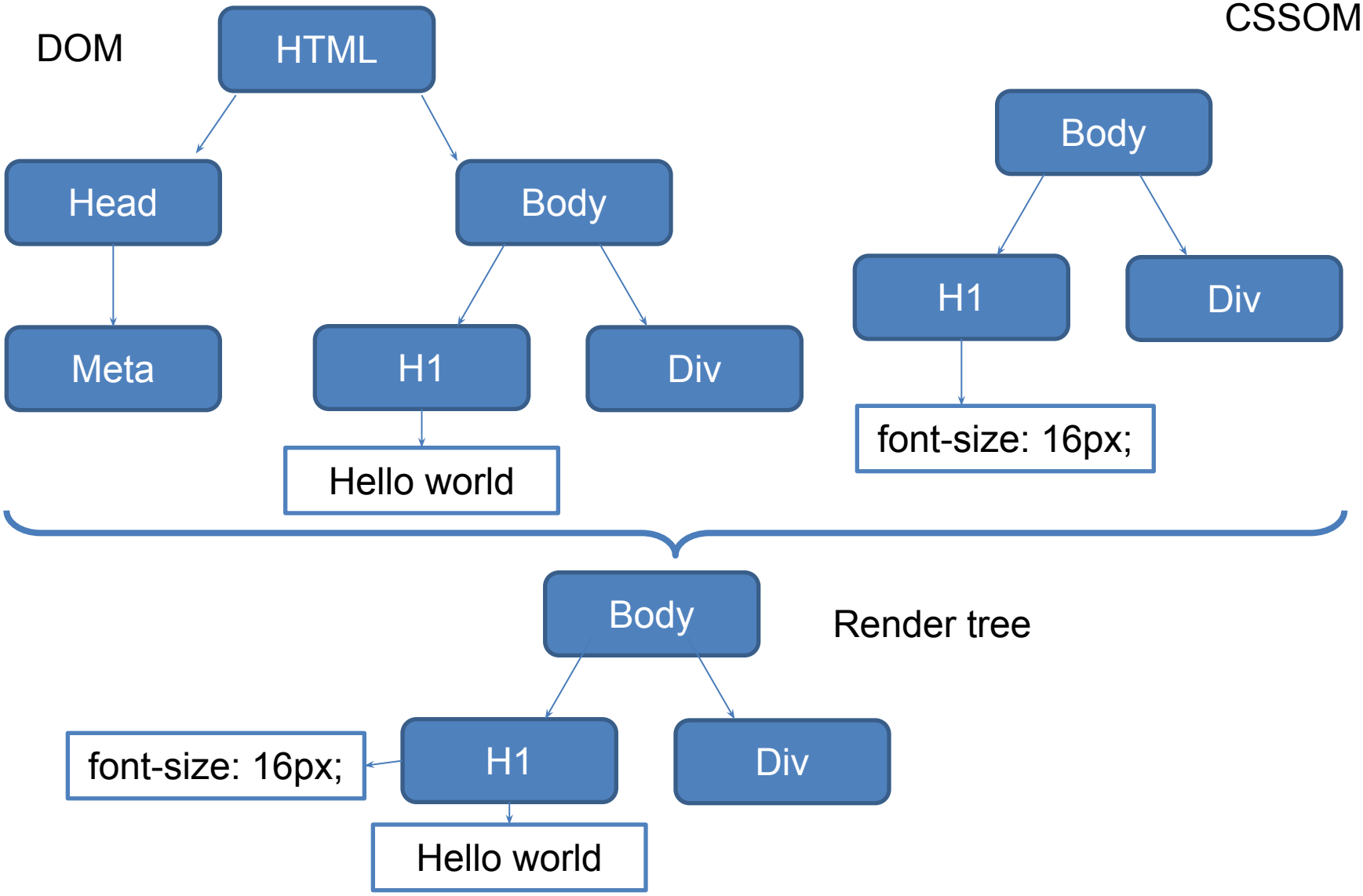
```
.green-background {
    background-color: green;
    width: 50%;
    height: 100px;
}
```

РЕНДЕРИНГ HTML

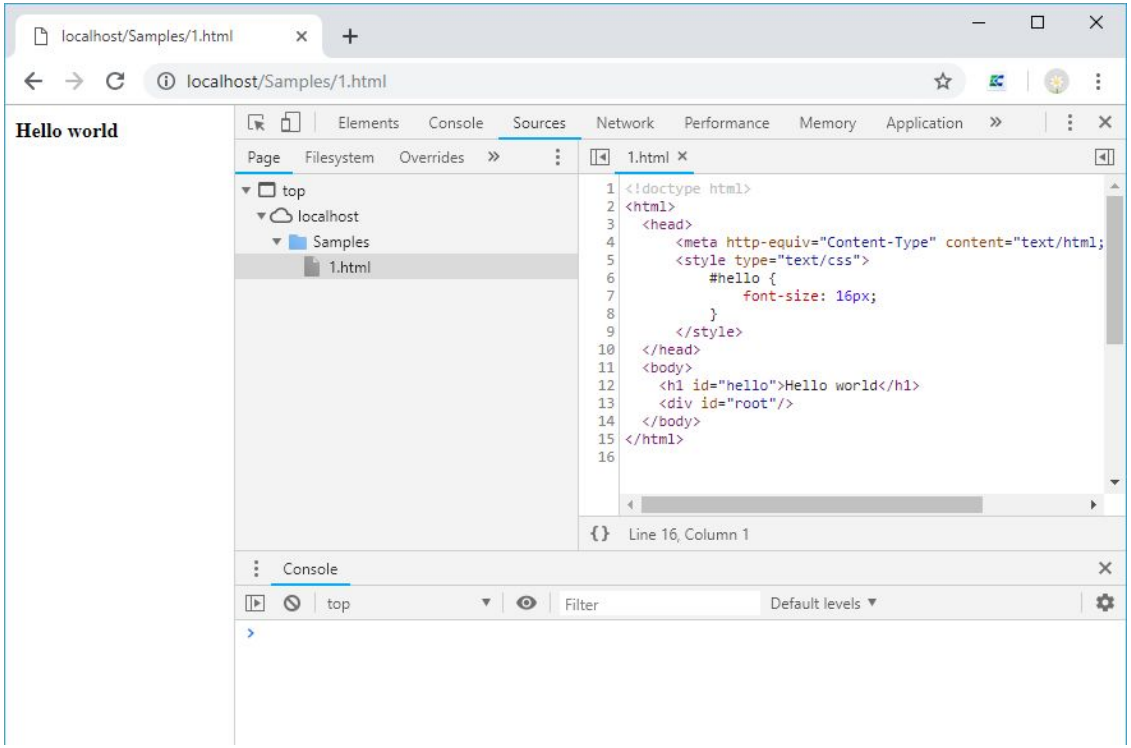
1. Из полученного от сервера HTML формируется **DOM** (Document Object Model).
2. Загружаются и распознаются стили, формируется **CSSOM** (CSS Object Model).
3. На основе DOM и CSSOM формируется дерево рендеринга (**Render tree**) — полный набор объектов рендеринга.
4. Для каждого элемента Render tree рассчитывается положение на странице — происходит **layout**. Браузеры используют поточный метод (flow).
5. Наконец, происходит отрисовка всего этого в браузере — **painting**.

Автоисправление

RENDER TREE


DOM
HTML
Head
Body
Meta
H1
Div
Hello world
CSSOM
Body
H1
Div
font-size: 16px;
Body
Render tree
font-size: 16px;
H1
Div
Hello world

ИНСТРУМЕНТЫ РАЗРАБОТЧИКА CHROME



1. Вкладка **Network** отображает в реальном времени процесс загрузки всех ресурсов, указанных в разметке страницы.
2. Вкладка **Console** отображает ошибки на странице и вывод с помощью **console.log(…)**

3. Вкладка **Elements** отображает текущее состояние страницы (Render Tree). Используется для непосредственного просмотра содержимого DOM.
4. Вкладка **Sources** отображает все ресурсы, загруженные вместе с HTML – страницей. Используется для отладки js-кода, установки **breakpoints**.

DOM

Document Object Model — это программный интерфейс, другими словами - инструмент, с помощью которого JavaScript видит содержимое HTML-страницы и состояние браузера, и может управлять ими.

Для обеспечения взаимной и обратной совместимости специалисты W3C классифицировали эту модель по уровням.
Спецификации всех уровней объединены в общую группу, носящую название «**W3C DOM**».

Каждый DOM-узел (элемент) является объектом, соответствующим синтаксису Javascript и содержащим все свойства, определяющие его вид и поведение.

Свойства DOM-узлов:

- tagName, style, innerHTML, **innerText**, className и пр. атрибуты, либо определенные в исходной HTML – разметке, либо установленные посредством javascript
- Обработчики событий (начинаются на on..)

ОБЪЕКТЫ И СОБЫТИЯ
DOM

В общем случае, события различного вида выделяют, основываясь на объекте, который вызывает данное событие:

- Объект **window** группирует все свойства браузера
- Объект **document** группирует все свойства HTML документа, обеспечивает пользовательское взаимодействие
- Объект в DOM

- Глобальный объект **window** вызывает событие '**load**', когда страница закончила рендеринг, что все ресурсы были скачаны и применены
- Объект **document**, представляющий HTML-документ, вызывает событие, '**DOMContentLoaded**', когда документ закончил загрузку,
- Каждый объект узла DOM, такой как div или button, вызывающий событие, называемое '**click**', когда пользователь нажимает кнопку мыши, пока ее указатель находится поверх узла DOM на HTML-странице.

JAVASCRIPT – СРЕДСТВО ДОСТУПА К DOM

Исключительная «фича» javascript – поддержка всеми браузерами.

Ключевые особенности языка:

- **Однопоточность!**
- Прототипно – ориентированный язык
- Слабая типизация, можно сказать, ее нет
- **Функции являются объектами**, по сути, данными

Для работы с DOM при помощи Javascript используются глобальные объекты

- **document**
- **window**

Объект **window** является глобальной областью видимости (this) в каждом блоке <script>.

Таким образом следующие записи являются идентичными:

```
document.getElementById('counter');
window.document.getElementById('counter');
```

ОБРАБОТКА СОБЫТИЙ
DOM

Вместо определения значений атрибута `onclick` можно добавлять элементам реакции на события. Например, вместо

```
<body onload="alertMessage();">
```

можно написать:

```
window.onload = alertMessage;
window.addEventListener('load', alertMessage, false);
```

Ключевые особенности языка при обработке событий:

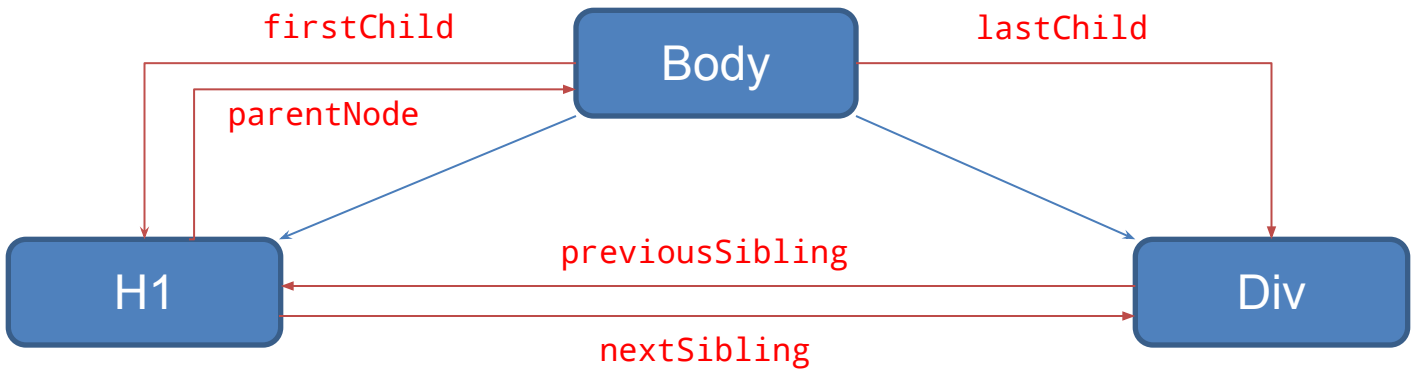
- **Однопоточность**
- **Замыкания**
- **Функции являются объектами, по сути, данными**

```
function first(){
  setTimeout (function(){
    console.log(1);
  }, 1000 );
 console.log(2);
}
```

ПОИСК И НАВИГАЦИЯ В DOM

Есть несколько способов получить объект, представляющий узел, через глобальный объект `document` или уже имея ссылку на другой узел

1. Если узел имеет уникальный идентификатор (атрибут `id`), то узел можно найти с помощью метода `document.getElementById(id)`.
2. Можно найти массив элементов с заданными тегами с помощью метода `document.getElementsByTagName(tag)`.
3. Можно перейти от узла к его непосредственным потомкам `node.firstChild, node.lastChild` или к предку `node.parentNode`.
4. Можно перейти от узла к его соседям `node.nextSibling`, `node.previousSibling`.



ВНЕСЕНИЕ ИЗМЕНЕНИЙ В DOM

Следующие методы применимы ко всем элементам DOM:

1. `element.appendChild(node)` – добавление нового узла в конец списка "детей".
2. `element.insertBefore(newNode, oldNode)` – добавление нового узла в список детей перед заданным.
3. `element.removeChild(node)` – удаление указанного узла из списка "детей".
4. `element.replaceChild(newNode, oldNode)` – замена в списке "детей" существующего элемента на новый.

Новый элемент (атрибут, текстовый узел) можно создать с помощью следующих методов:

1. `document.createElement(tag)` – создание нового элемента с заданным тегом.
2. `document.createAttribute(name)` – создание нового атрибута с заданным именем.
3. `document.createTextNode(data)` – создание текстового узла.

ИЗМЕНЕНИЯ В DOM

Repaint

В случае изменения стилей элемента, не влияющих на его размеры и положение на странице (например, цвет), браузер просто отрисовывает его заново, с учётом нового стиля — происходит repaint (или restyle).

Reflow

Если же изменения затрагивают содержимое, структуру документа, положение элементов — происходит reflow (или relayout). Причинами таких изменений обычно являются:

- Манипуляции с DOM (добавление, удаление, изменение);
- Изменение содержимого
- Изменение CSS
- Манипуляции с окном браузера — изменения размеров, прокрутка;

РЕАЛИЗАЦИЯ DOM В ВЕБ-БРАУЗЕРАХ

Учитывая существование различных реализаций DOM в веб-браузерах, среди программистов бытовала привычка сперва проверять работоспособность тех или иных возможностей DOM для конкретного браузера и только потом использовать их.

Код ниже иллюстрирует способ проверки на поддержку стандартов «W3C DOM».

```
// Если метод getElementById доступен, можно делать вызов.
if (document.getElementById) {
  var counter = document.getElementById("counter")
}
```

Контрольный вопрос: Почему работает конструкция

```
if (document.getElementById)
```

JQUERY

Jquery - кроссбраузерная библиотека, упрощающая взаимодействие с DOM средствами Javascript

```
$(".some-class") вернет все элементы с классом some-class.
$("#root")   вернет элемент с идентификатором root.


// Изменит содержимое элемента с идентификатором root
$("#root").html(<p>Новое содержимое</p>)
```

jQuery применяют в 96,5% всех сайтов (иссл. 2016 г.)

```
// Это работает аналогично getElementById, в любом браузере
var counter = jQuery('#counter');

// Функция $ является кратким синонимом jQuery
var counter = $('#counter');
```

ПРОБЛЕМА DOM

Главная проблема DOM

Он изначально не был рассчитан для создания динамического пользовательского интерфейса.
Мы можем работать с ним, используя JavaScript и библиотеки наподобие jQuery, но их использование не решает проблем с производительностью.

Взглянем на современные социальные сети, такие как Twitter, Facebook.
После небольшого скроллинга, мы будем иметь в DOM десятки тысяч узлов.
Для примера, попробуем переместить 1000 div-блоков на 5 пикселей влево.

VIRTUAL DOM

Решение проблем с производительностью DOM

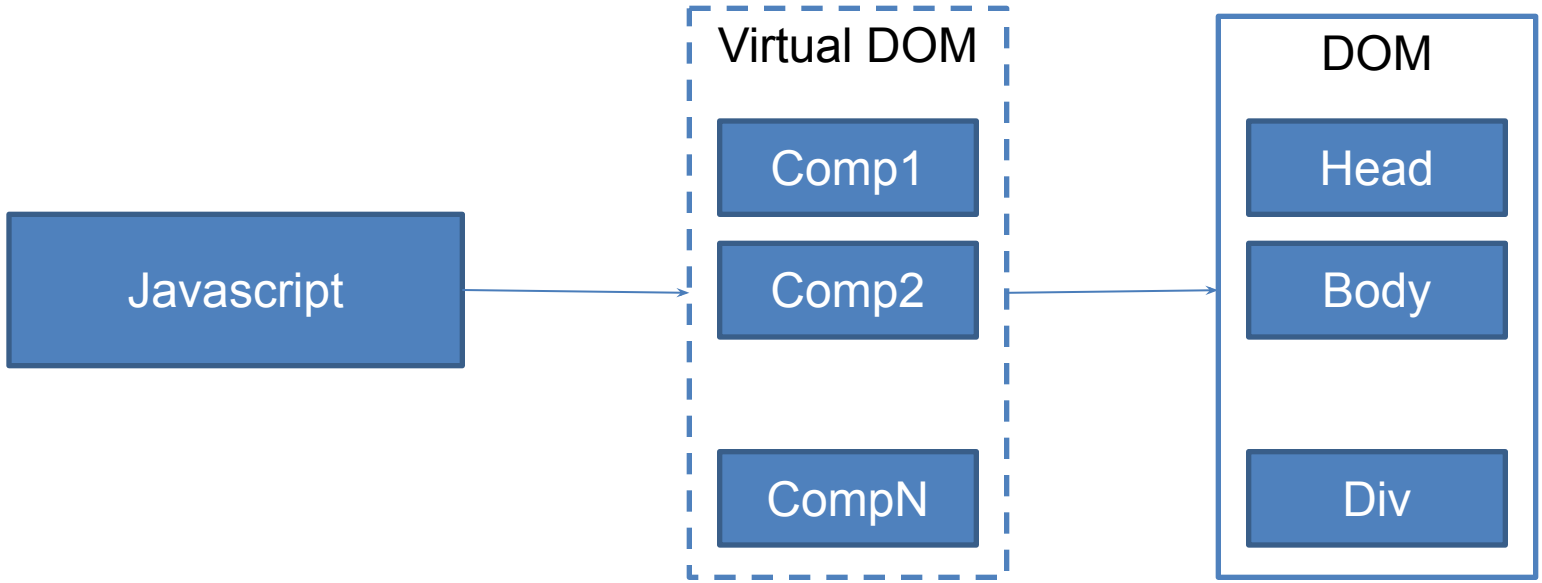
1. В настоящее время W3C работает над новым стандартом Shadow DOM. Shadow DOM — описывает метод объединения нескольких DOM-деревьев в одну иерархию и как эти деревья взаимодействуют друг с другом в пределах документа, что позволяет лучше скомпоновать DOM.

2. Другой вариант заключается в использовании подхода с **Virtual DOM**. Virtual DOM не является стандартом, и в конечном итоге мы по-прежнему взаимодействуем с DOM, но делаем это как можно реже и более эффективно.

REACT JS

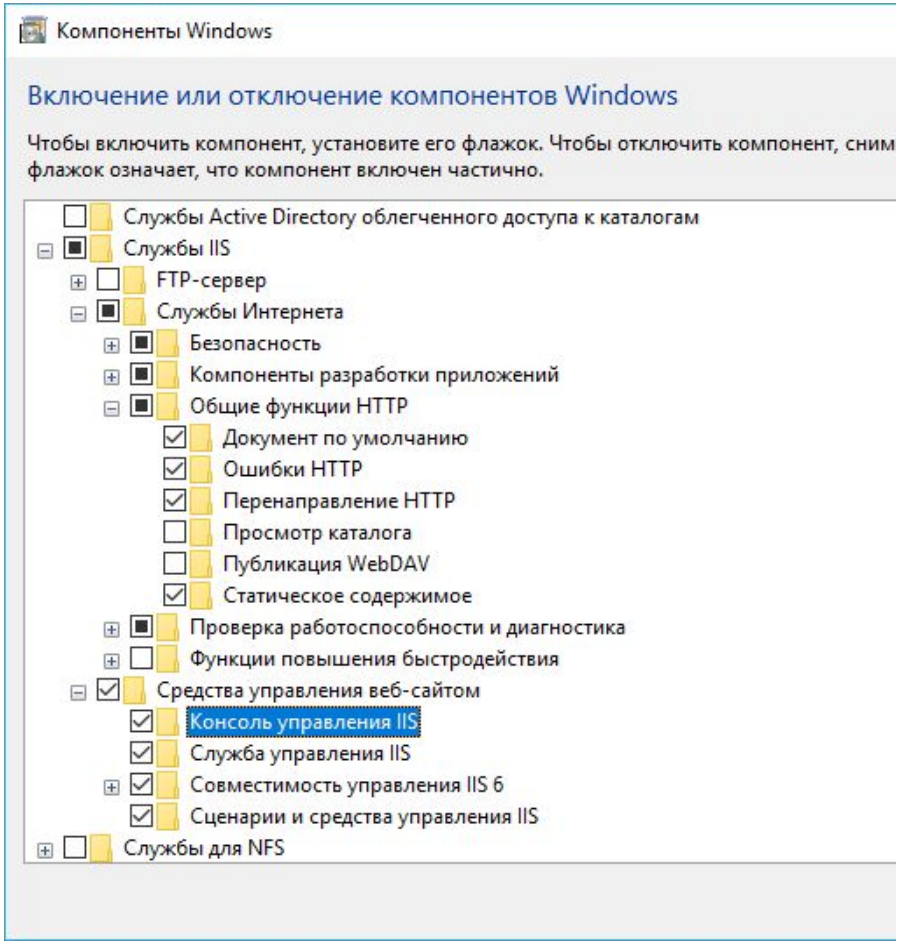
Браузер не «знает» про виртуальный DOM – это лишь шаблон программирования, когда «виртуальное» представление пользовательского интерфейса хранится в памяти и синхронизируется с «реальным» DOM с помощью библиотеки reactJS.

Другими словами, это Javascript – объектное представление DOM, которое мы можем изменять так часто, как нам нужно. Изменения, внесенные в этот объект, затем сопоставляются, а изменения в реальном DOM производятся намного реже.



УСТАНОВКА IIS

Для работы простого веб-сайта достаточно установки следующих компонентов (Оснастка «Включение или отключение компонентов Windows»)



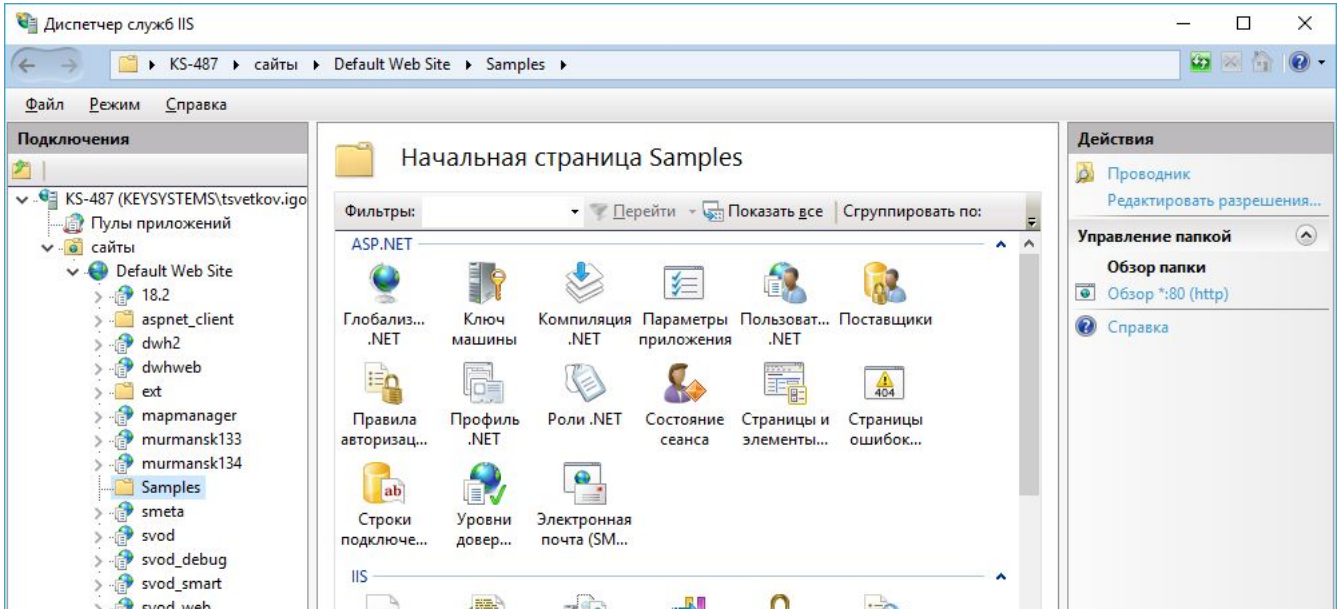
*HTML может быть открыт по разным протоколам

file:///C:/inetpub/wwwroot/Samples/1.html

http://localhost/Samples/1.html

НАСТРОЙКА IIS

Оснастка для управления веб-сайтами и приложениями – **Диспетчер служб IIS** (Панель управления - Администрирование)



Рабочим каталогом по умолчанию для веб-сервера IIS является ***C:\inetpub\wwwroot***

1. Помещаем папку Samples внутрь wwwroot: C:\inetpub\wwwroot\Samples
2. Открываем в браузере ссылку http://localhost/Samples/find.html

СПАСИБО ЗА ВНИМАНИЕ!

ООО «Кейсистемс»
428000 Чувашская республика
г. Чебоксары, ул. И. Яковлева, д. 3
тел./факс: (8352) 323-323
www.keysystems.ru