Окружение frontend-приложений

Борис Ребров

ЦРИТО
Центр Развития
ИТ-Образования

МФТИ

@mail.ru group

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях

Что сегодня?

- Окружение в котором работает web-приложение
- Инструменты для исследования этого окружения
- Стандарты и их развитие

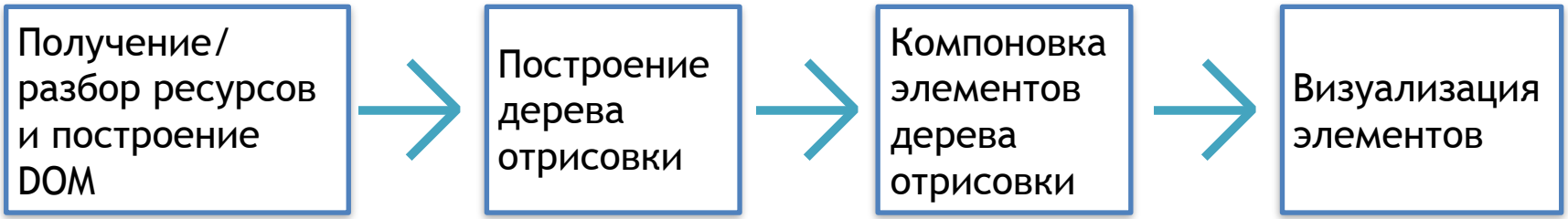
Из чего сделан браузер

- Парсинг и рендеринг документов
- Интерпретация скриптов
- Инструменты разработки

https://chromium.googlesource.com/

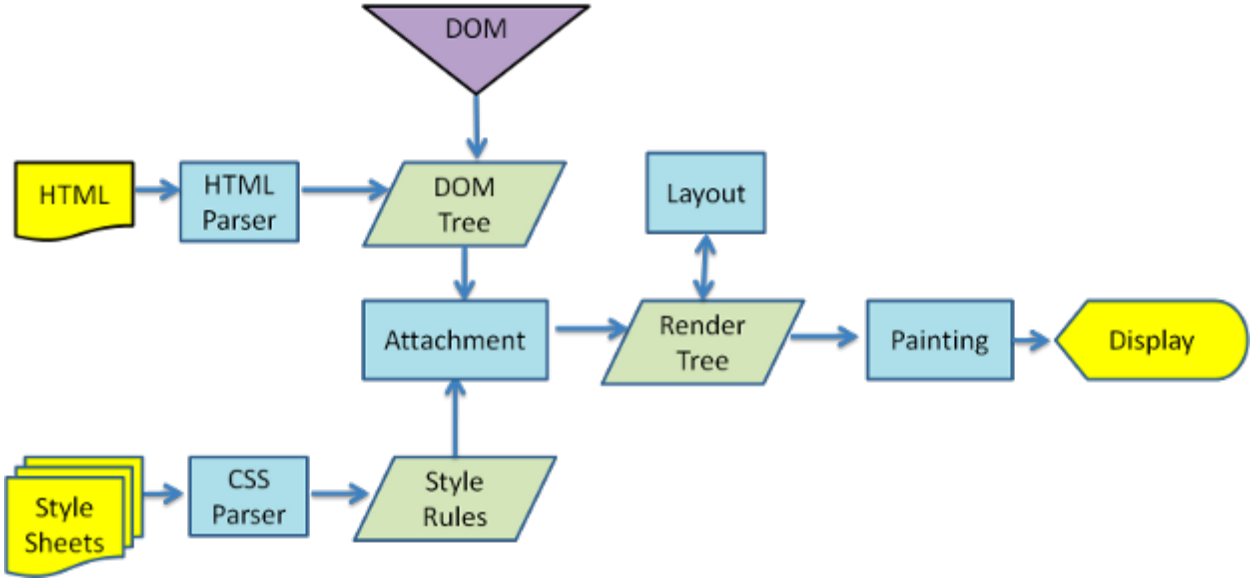
https://developers.google.com/web/updates/2018/

Жизненный цикл страницы



https://html5rocks.appspot.com/en/tutorials/internals/howbrowserswork/

Жизненный цикл страницы



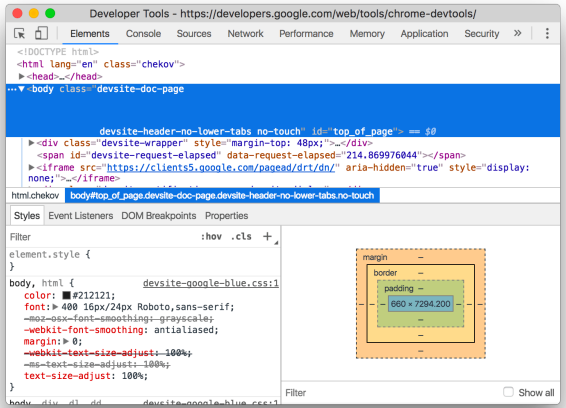
https://html5rocks.appspot.com/en/tutorials/internals/howbrowserswork/

Жизненный цикл страницы

1. Пользователь взаимодействует с документом
2. Все хотят 60fps
3. Не все изменения одинаково полезны

Chrome Developer Tools

- Инспектор DOM и CSS
- Дебаггер javascript
- Инспектор сетевого взаимодействия
- Профайлер



https://developers.google.com/web/tools/chrome-devtools/

JavaScript

- Клиентский
- Прототипно-ориентированый
- Динамическая типизация
- Лексическая область видимости
- Самый популярный язык в мире*
- wat-синтаксис

http://learn.javascript.ru/

* по данным исследований GitHub и StackOverflow

JavaScript – замыкания

```
1. function bind(func, context){
2.  return function() {
3.   return func.apply(
4.      context,
5.      arguments
6.     );
7.  }
8. }
```

JavaScript – объекты

- Все – object
- Объект – фактически ассоциативный массив или словарь

```
1. (‘test’).length; // 4
2. (9).constructor // Number
3. (function my () {}).name // «my»
```

- Все глобальные переменные и функции – свойства глобального объекта

```
1. myVar = 1; // без var
2. console.log(window.myVar); // 1
```

JavaScript – объекты

- delete работает только для свойств объектов

```
1. myVar = 1; // без var
2. console.log(window.myVar); // 1
3. delete myVar // true
4. console.log(window.myVar); //undefined
```

- У объекта есть прототип, а в прототипе – ссылка на конструктор

```
1. [].constructor // Array
2. [].constructor.prototype // {constructor: Array, concat: …}
3. [].constructor.prototype == Object.getPrototypeOf([]) // true
```

JavaScript – прототипы

- Оператор new вызывает функцию в контексте нового объекта

```
1.  function MyConstructor () {
2.      this.myProperty = 1;
3.      return this.myProperty;
4.  };

5.  new MyConstrustor;

7.  // this = {};
8.  // this.__proto__ = MyConstructor.prototype

10. this.myProperty = 1;

12. // return this;
13.
```

JavaScript – прототипы

- Поиск свойств осуществляется по цепочке прототипов

```
1.  function MyConstructor (...args) {
2.      this.push.apply(this, Array.from(args))
3.  };
4.  MyConstructor.prototype = Object.create(Array.prototype);
5.  var test = new MyConstructor(1,2,3);
6.  test.join(‘,’); // ‘1,2,3’
7.  // test.__proto__.__proto__.join
8.  test.hasOwnProperty(1) // true
9.  // test.__proto__.__proto__.__proto__.hasOwnProperty


12. MyConstructor.prototype.join = function join (...args) {
13.   console.log('join');
14.   return Array.prototype.join.apply(this, args);
15. }
```

JavaScript – расширенные возможности

- get/set/defineProperty
- let/const
- map/reduce/filter
- Деструктуризация

```
let [first, second] = ‘Hello world!’.split(‘ ‘);
//first - ‘Hellow’, second - ‘world’
let [first, second, …rest] = ‘Hello world! My name is JS’.split(‘ ‘);
//first - ‘Hellow’, second - ‘world’, rest - массив
```

- Стрелочные функции

```
element.addEventListener(‘click’, event => event.preventDefault());
[‘one’, ‘two’].map(item => item.toUpperCase());
```

JavaScript – классы

```
1.  class Animal {
2.    constructor(name) {
3.      this.name = name;
4.    }

6.    walk() {
7.      alert("I walk: " + this.name);
8.    }
9.  }

11. class Rabbit extends Animal {
12.   walk() {
13.     super.walk();
14.     alert("...and jump!");
15.   }
16. }

18. new Rabbit("Вася").walk();
```

JavaScript – модули

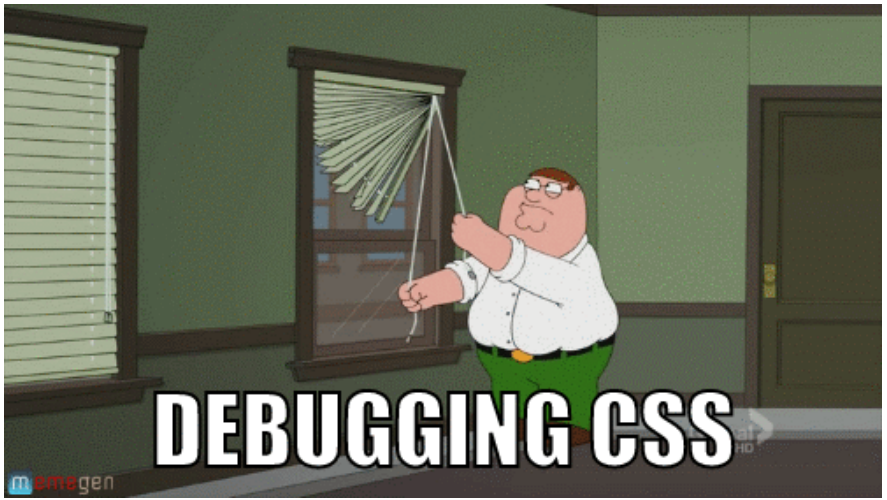
```
export class Test {
  constructor(name) {
    this.name = name;
  }
};

export function sayHi() {
  alert("Hello!");
};
```

```
import * as imports from ‘./imports.js’;
imports.sayHi();
const testInstance = new imports.Test;
```

HTML/CSS

- Нестрогое подмножество xml
- Браузеры прощают очевидные ошибки
- Куча боли



HTML

```
<!doctype html>
<html>
    <head>
        <meta charset='utf-8' />
        <title>Hello world!</title>
    </head>
    <body>
        <h1>Статья</h1>
        <article>
            <p>Текст статьи.</p>
            <svg>
                <circle r="50" cx="50" cy="50" fill=«green»/>
            </svg>
        <article>
        <footer>
            «Подвал» страницы
        </footer>
    </body>
</html>
```

CSS

- Атрибут style

```
<body>
    <div style='border: 1px solid red'>text</div>
</body>
```

- Тег style

```
<style>
    div {
        border: 1px solid red;
    }
</style>
```

- css файл

```
<link rel='stylesheet' type='text/css' href='style.css'>
```

CSS селекторы

- Простые селекторы

1. `.class`
2. `#id`
3. `tagname`
4. `*`
5. `[attribute]`

- Комбинаторы

1. `div p`
2. `div > p`
3. `div + p`
4. `div ~ p`
5. `col || td`

CSS селекторы

- Примеры

```
div {
    color: black;
}
span {
    color: reb;
}
```

```
<body>
    <div>
        <span>text</span>
        <div>
            text in block
        </div>
    </div>
</body>
```

CSS селекторы

- Примеры

```
1. body * {
2.     color: black;
3. }
4. span {
5.     color: reb;
6. }
```

```
1. <body>
2.     <div>
3.         <span>text</span>
4.         <div>
5.             text in block
6.         </div>
7.     </div>
8. </body>
```

CSS селекторы

- Примеры

```
1. body div span {
2.     color: black;
3. }
4. span + div {
5.     color: reb;
6. }
```

```
1. <body>
2.     <div>
3.         <span>text</span>
4.         <div>
5.             text in block
6.         </div>
7.     </div>
8.     <span>text</span>
9. </body>
```

CSS селекторы

- Псевдо-классы

1. `:first-of-type`
2. `:visited`
3. `:checked`
4. `:hover`

- Псевдо-элементы

1. `::before`
2. `::after`
3. `::first-line`

CSS селекторы

- Примеры

```
1. body span:first-of-type {
2.     color: black;
3. }
4. span ~ span {
5.     color: reb;
6. }
```

```
1. <body>
2.     <div>
3.         <span>text</span>
4.         <div>
5.             text in block
6.         </div>
7.         <span>text</span>
8.     </div>
9.     <span>text</span>
10. </body>
```

CSS селекторы

- Сопоставление селекторов – дорогая операция
- Чем больше вложенность и меньше конкретики – тем дороже
- BEM
- CSS-modules

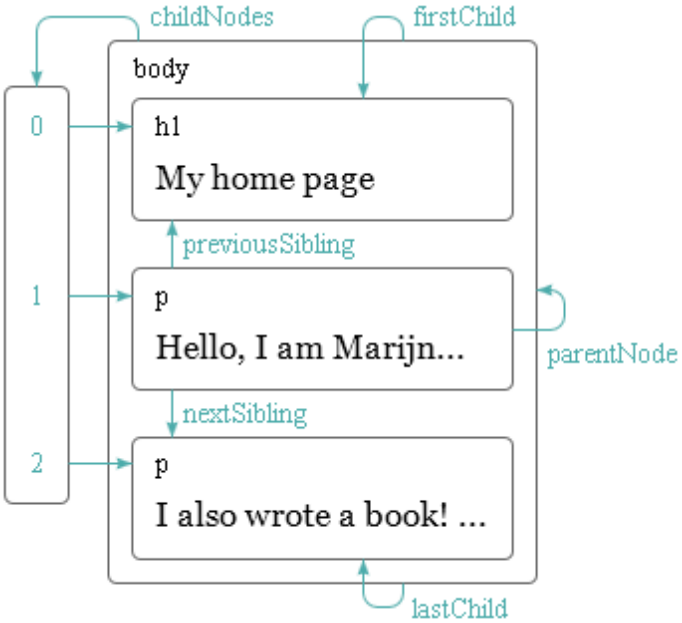
CSS и js

```
<body>
    <style>
        div {
            display: inline;
        }
    </style>
    <div style=‘border: 1px solid red’>text</div>
</body>
```

```
var myDiv = document.body.querySelector('div:first-of-type');
myDiv.style.borderColor; // ‘red’
myDiv.style.display; // ‘’
getComputedStyle(myDiv[,pseudo]).display; // ‘inline’
```

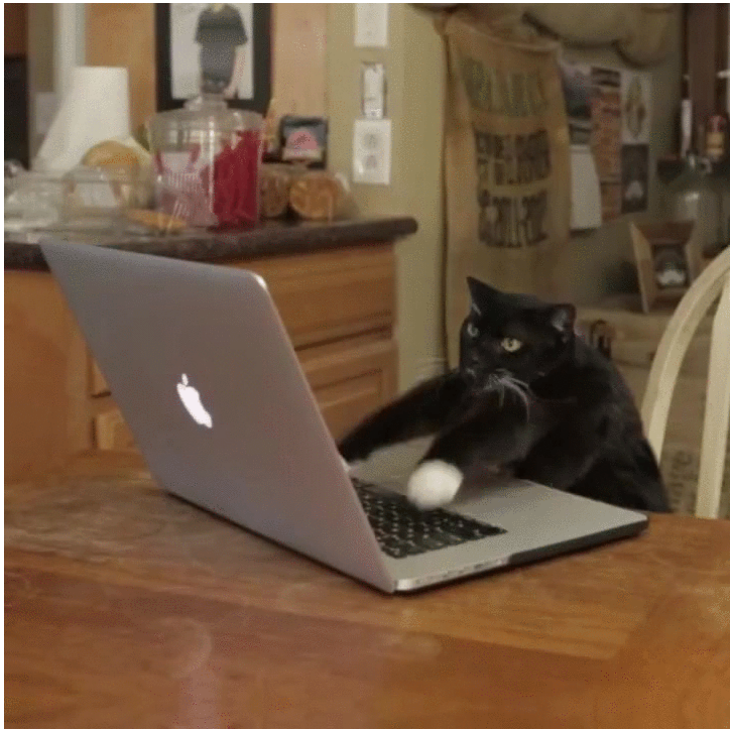
DOM

- Ноды иерархически связаны
- Ноды – те же объекты
- Дает доступ к свойствам элементав
- Обработчики событий
- Модификация документа



WebComponents

Пишем код



Домашнее задание № 2

- Ознакомиться с документацией по ссылкам
- Доработать страницу формы отправки сообщений
- Сохранять сообщения/данные пользователя в localStorage и восстанавливать их

Срок сдачи

- ~ 8 октября

Спасибо за внимание!