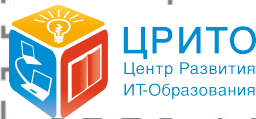




Окружение frontend-приложений

Борис Ребров



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



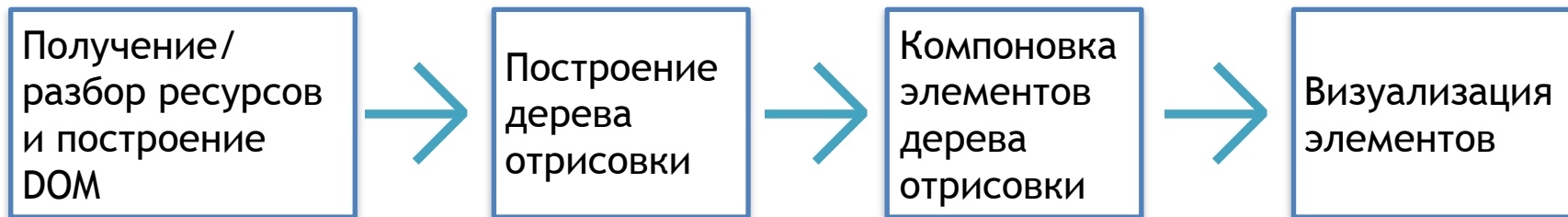
- Парсинг и рендеринг документов
- Интерпретация скриптов
- Инструменты разработки



<https://chromium.googlesource.com/>

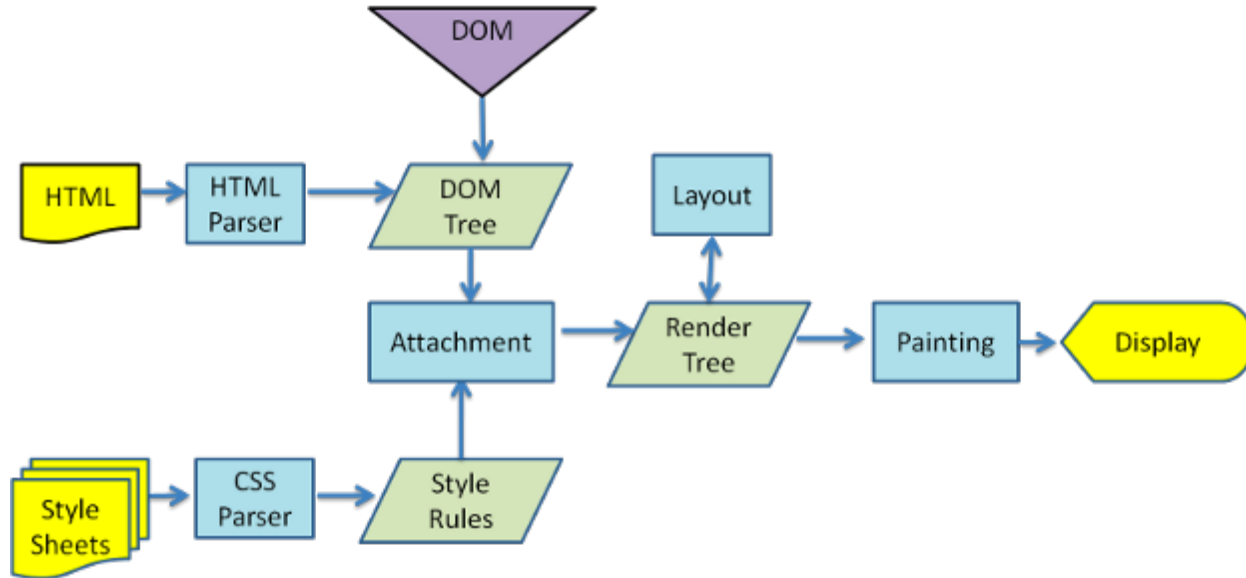
<https://developers.google.com/web/updates/2018/>

Жизненный цикл страницы



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

Жизненный цикл страницы

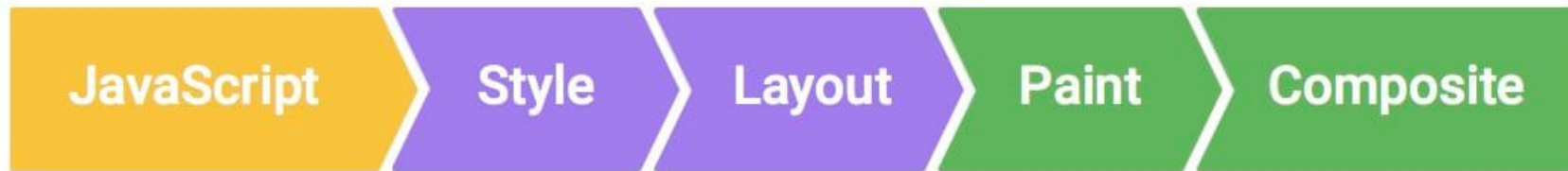


<https://html5rocks.appspot.com/en/tutorials/internals/howbrowserswork/>

Жизненный цикл страницы



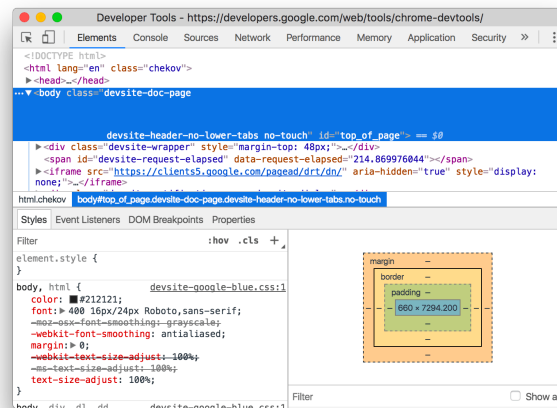
1. Пользователь взаимодействует с документом
2. Все хотят 60fps
3. Не все изменения одинаково полезны



Chrome Developer Tools



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



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



- Клиентский
- Прототипно-ориентированный
- 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. }
```



- 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`



- 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
```



- Оператор new вызывает функцию в контексте нового объекта

```
1. function MyConstructor () {  
2.     this.myProperty = 1;  
3.     return this.myProperty;  
4. };  
  
5. new MyConstructor;  
  
7. // this = {};  
8. // this.__proto__ = MyConstructor.prototype  
  
10. this.myProperty = 1;  
  
12. // return this;  
13.
```



- 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. }
```



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

```
1. let [first, second] = 'Hello world!'.split(' ');
2. //first - 'Hellow', second - 'world'
3. let [first, second, ...rest] = 'Hello world! My name is JS'.split(' ');
4. //first - 'Hellow', second - 'world', rest - массив
```

- Стрелочные функции

```
1. element.addEventListener('click', event => event.preventDefault());
2. ['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



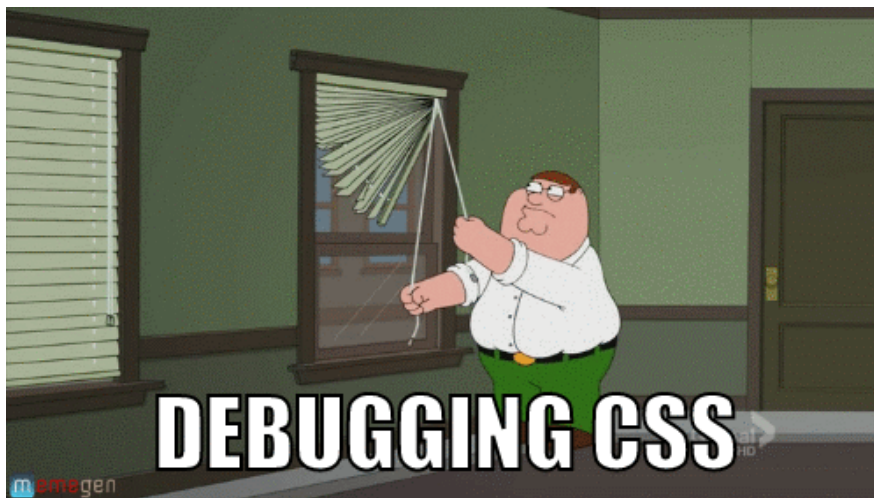
```
1. export class Test {  
2.   constructor(name) {  
3.     this.name = name;  
4.   }  
5. };
```

```
7. export function sayHi() {  
8.   alert("Hello!");  
9. };
```

```
1. import * as imports from './imports.js';  
2. imports.sayHi();  
3. const testInstance = new imports.Test;
```



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





```
1. <!doctype html>
2. <html>
3.     <head>
4.         <meta charset='utf-8' />
5.         <title>Hello world!</title>
6.     </head>
7.     <body>
8.         <h1>Статья</h1>
9.         <article>
10.            <p>Текст статьи.</p>
11.            <svg>
12.                <circle r="50" cx="50" cy="50" fill=«green»/>
13.            </svg>
14.        </article>
15.        <footer>
16.            «Подвал» страницы
17.        </footer>
18.    </body>
19. </html>
```



- Атрибут style

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

- Тер style

1. `<style>`
2. `div {`
3. `border: 1px solid red;`
4. `}`
5. `</style>`

- css файл

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



- 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`



- Псевдо-классы

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

- Псевдо-элементы

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

CSS селекторы



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

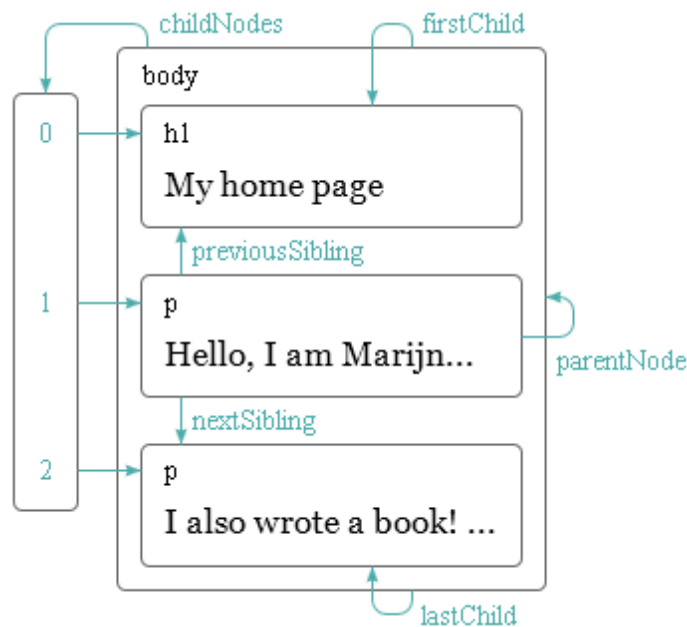


```
1. <body>
2.     <style>
3.         div {
4.             display: inline;
5.         }
6.     </style>
7.     <div style='border: 1px solid red'>text</div>
8. </body>
```

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




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







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!