Введение во фронтенд
Основы JavaScript

Лекция 2

Михаил Привер

Обо мне

Михаил Привер
Поиск Mail.Ru
Пульс Mail.Ru

Важно!

- Отметиться на портале
- Оставить обратную связь

О чем эта лекция?

- Как работает браузер
- DevTools
- Основы JS, CSS и HTML

Браузеры







Из чего состоит браузер

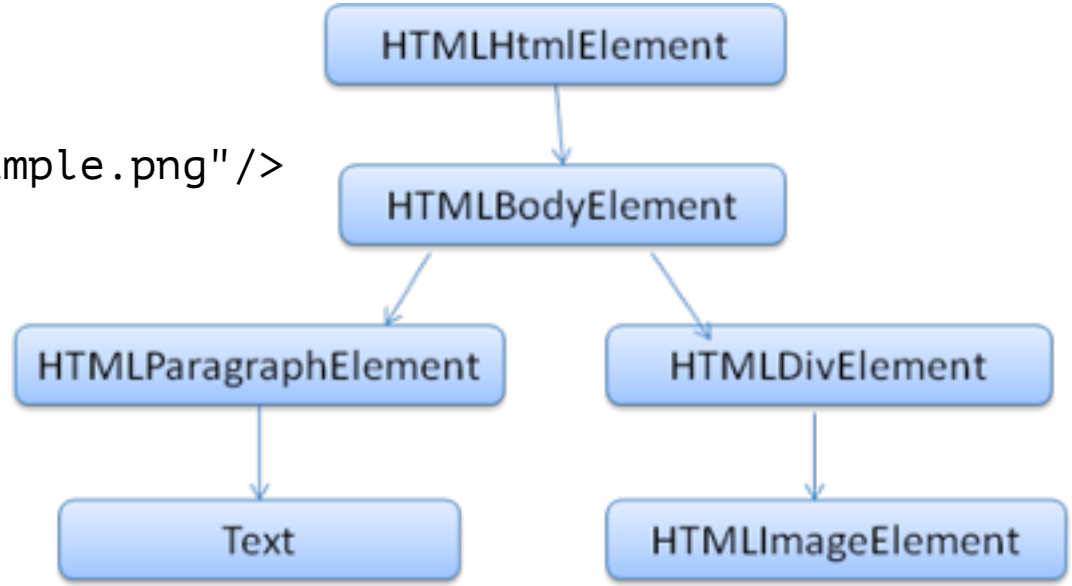
- Браузерный движок
  (преобразование HTML-документов в интерактивное изображение)
- Движок JavaScript
  (интерпретация и выполнение скриптов)
- Инструменты разработки

Как происходит рендеринг

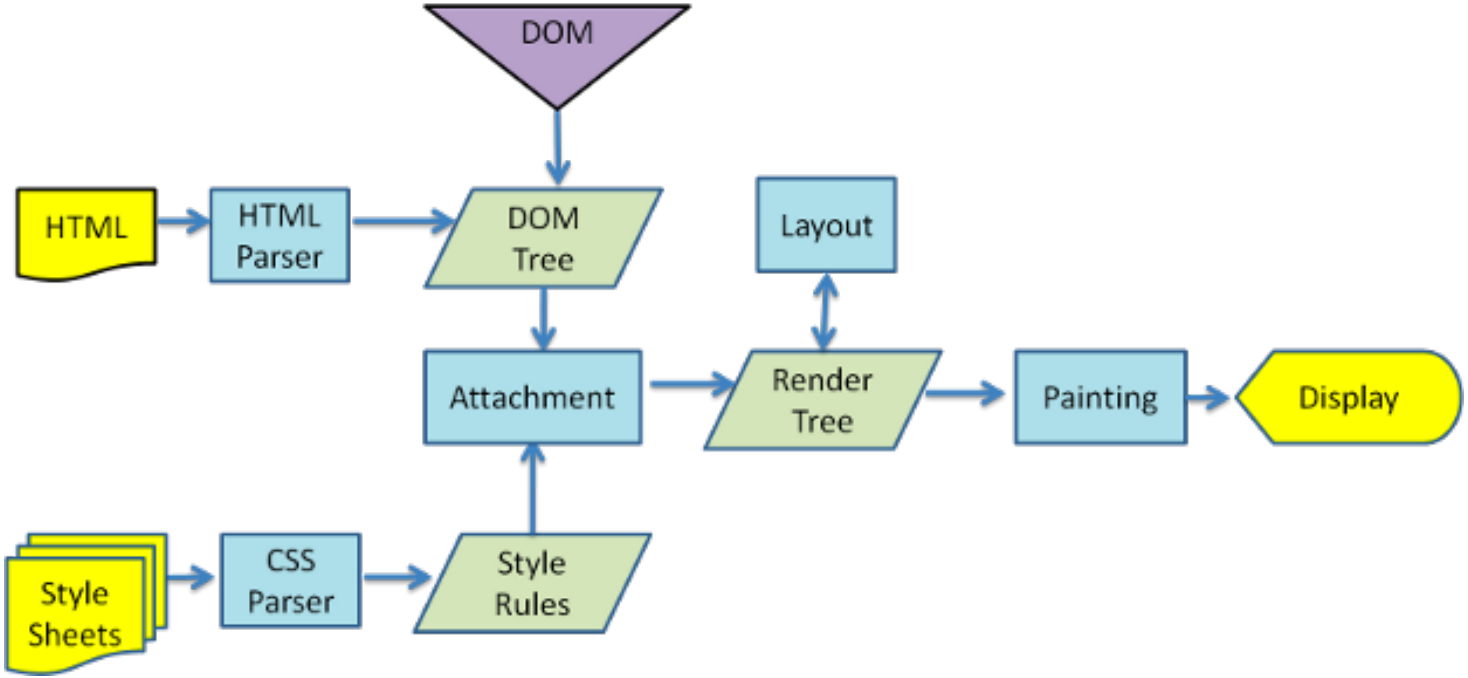
https://html5rocks.appspot.com/en/tutorials/internals/howbrowserswork/

DOM

```
1. <html>
2.     <body>
3.         <p>
4.             Hello World
5.         </p>
6.         <div>
7.             <img src="example.png"/>
8.         </div>
9.     </body>
10.</html>
```

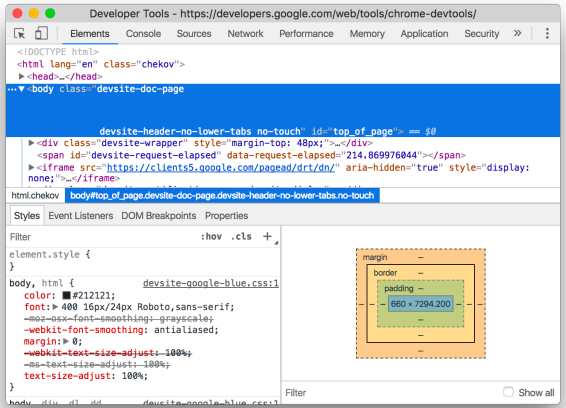


Рендеринг в Chrome



Chrome Developer Tools

- Инспектор DOM и CSS
- Дебаггер JavaScript
- Инспектор сетевого взаимодействия
- Профилировщик



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

JavaScript

- Клиентский
- Функции первого класса
- Лексическая область видимости
- Прототипно-ориентированый
- Динамическая типизация
- Самый популярный язык в мире
  (по данным исследований GitHub и StackOverflow)

http://learn.javascript.ru/

JavaScript – объекты

- 6 примитивных типов:
  number, string, boolean, null, undefined, symbol
- Все остальное — объекты
- Оператор typeof
- Функции обертки: Number, String, Boolean

JavaScript – преобразование типов

- **Строковое: String(value)**
- **Численное: Number(value)**

```
undefined => NaN
null => 0
true / false => 1 / 0
'' => 0
' 10 кошек' => 10
'собака' => NaN
```

- **Логическое: Boolean(value)**

```
0, null, undefined, NaN, '' => false
остальное => true
```

JavaScript — методы объектов

- Функции, которые находятся в объекте в качестве его свой ств, называются «методами».
- Методы позволяют объектам «действовать»: object.doSomething().
- Методы могут ссылаться на объект через this.

JavaScript — преобразование объектов к примитивам

- Функции, которые находятся в объекте в качестве его свой ств, называются «методами».
- Методы позволяют объектам «действовать»: object.doSomething().
- Методы могут ссылаться на объект через this.

JavaScript — операторы

- Условные операторы: if, '?'
- Циклы while и for
- Конструкция "switch"

JavaScript — замыкания

```
1. var i = 10;
2. var array = [];

4. while (i--) {
5.     array.push(function() {
6.         return i + i;
7.     });
8. }

10. console.log([
11.     array[0](),
12.     array[1](),
13. ]);
```

JavaScript — оператор new

Оператор new вызывает функцию в контексте нового объекта

```
1.  function User(name) {
2.      this.name = name;
3.  };

4.  var user = new User("Вася");

6.  // this = {};
7.  // this.__proto__ = User.prototype

9.  this.name = name;

11. // return this;
```

JavaScript — прототипное наследование

- В JavaScript все объекты имеют скрытое свойство [[Prototype]], которое является либо другим объектом, либо null
- Если мы хотим прочитать свойство obj или вызвать метод, которого не существует у obj, тогда JavaScript попытается найти его в прототипе.
- Операции записи/удаления работают непосредственно с объектом, они не используют прототип
- Свойство F.prototype (не путать с [[Prototype]]) устанавливает [[Prototype]] для новых объектов при вызове new F()

JavaScript — классы

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

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

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

18. new Rabbit("Вася").walk();
```

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 — модули

```
// say.js
export function sayHi() {
    console.log("Hello!");
};

export function sayBye() {
    console.log("Good Bye!");
};

export default function say(hiOrBye) {
    hiOrBye ? sayHi() : sayBye();
}
```

```
<!doctype html>
<script type="module">
  import say, { sayHi, sayBye } from './say.js';
  sayHi();
</script>
<script type="module">
  import * as say from './say.js';
  say.sayBye();
</script>

```

HTML и CSS

HTML

```
<!doctype html>
<html>
    <head>
        <meta charset=‘utf-8’ />
        <title>Hello world!</title>
        <link rel="stylesheet" href="./style.css">
    </head>
    <body>
        <h1>Статья</h1>
        <article>
            <p>Текст статьи.</p>
        <article>
        <footer>
            «Подвал» страницы
        </footer>
        <script src="./bundle.js"></script>
    </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" 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.selected || td`

CSS селекторы

Специфичность

- 1 если правило определено в атрибуте style, 0 иначе (= a)
- число атрибутов id в селекторе (= b)
- количество других атрибутов и псевдоклассов в селекторе (= c)
- количество имен элементов и псевдоэлементов в селекторе (= d)

```
1.  *               {}  /* a=0 b=0 c=0 d=0 -> specificity = 0,0,0,0 */
2.  li              {}  /* a=0 b=0 c=0 d=1 -> specificity = 0,0,0,1 */
3.  li::first-line  {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
4.  ul li           {}  /* a=0 b=0 c=0 d=2 -> specificity = 0,0,0,2 */
5.  ul ol+li        {}  /* a=0 b=0 c=0 d=3 -> specificity = 0,0,0,3 */
6.  h1 + *[rel=up]  {}  /* a=0 b=0 c=1 d=1 -> specificity = 0,0,1,1 */
7.  ul ol li.red    {}  /* a=0 b=0 c=1 d=3 -> specificity = 0,0,1,3 */
8.  li.red.level    {}  /* a=0 b=0 c=2 d=1 -> specificity = 0,0,2,1 */
9.  #x34y           {}  /* a=0 b=1 c=0 d=0 -> specificity = 0,1,0,0 */
10. style=""            /* a=1 b=0 c=0 d=0 -> specificity = 1,0,0,0 */
```

CSS селекторы

- Примеры

```
1. div {
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 * {
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

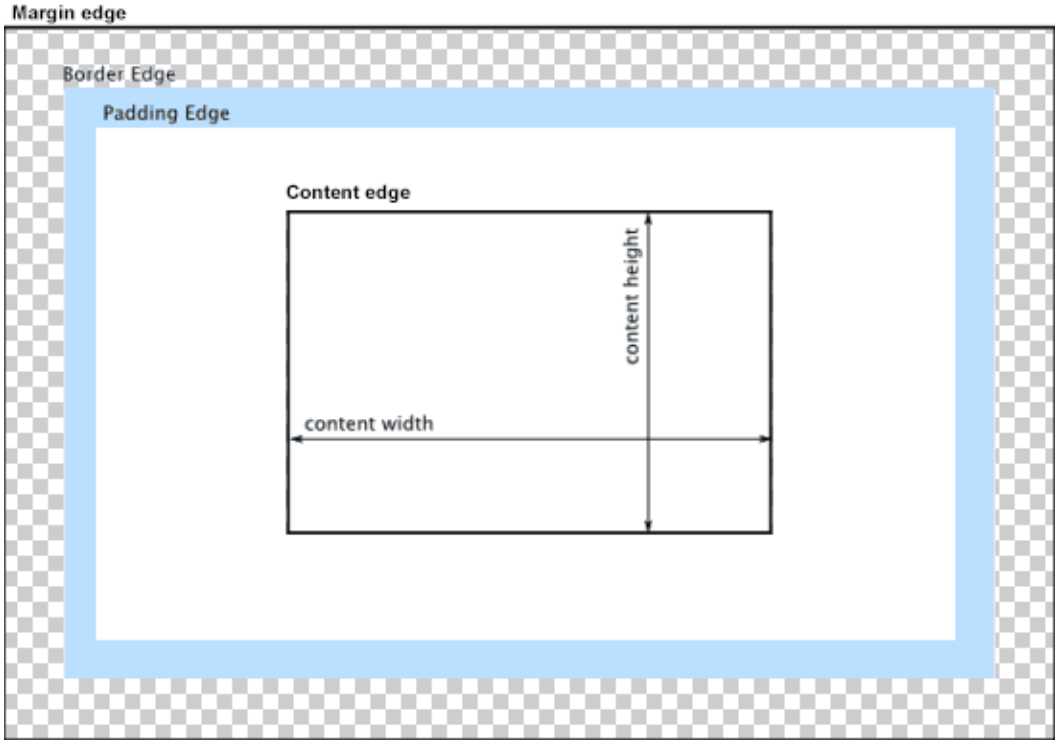
- Сопоставление селекторов – дорогая операция
- Чем больше вложенность и меньше конкретики – тем дороже
- BEM
- CSS-modules

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).display; // ‘inline’
```

Блоковая модель



Flexbox

https://codepen.io/enxaneta/full/adLPwv

WebComponents

- Custom Elements
- Shadow DOM
- <template> и <slot>

```
<p><slot name="my-text">My default text</slot></p>
```

```
<my-paragraph>
  <span slot="my-text">Let's have some different text!</span>
</my-paragraph>
```

```
<my-paragraph>
  <ul slot="my-text">
    <li>Let's have some different text!</li>
    <li>In a list!</li>
  </ul>
</my-paragraph>
```

Local Storage

- localStorage.getItem(key)
- localStorage.setItem(key, value)
- localStorage.removeItem(key)
- localStorage.clear()

Домашнее задание № 2

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

Срок сдачи

- ~ 14 октября

Спасибо за внимание!