



Лекция 4

CSS. Современные ВОЗМОЖНОСТИ

Мартин Комитски



ЦРИТО
Центр Развития
ИТ-Образования



План на сегодня

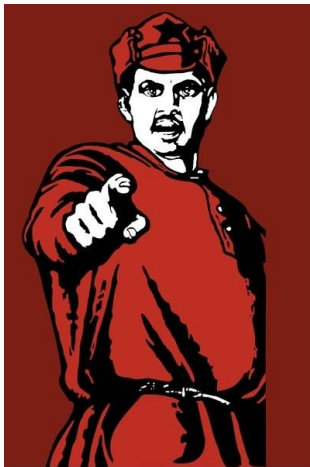


- CSS, боль
- Кто это придумал?
- **Возможности**
- Препроцессоры, инструменты
- **Разработка под мобильные устройства**
- Единицы измерения в CSS
- Область просмотра (viewport)
- Нативное взаимодействие
- ДЗ

Минутка бюрократии



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





CSS, боль



CSS

IS

AWESOME





Кто это придумал?



W3C (World Wide Web Consortium — Консорциум Всемирной паутины) — организация, разрабатывающая и внедряющая технологические стандарты для Всемирной паутины. Консорциум возглавляет **Тимоти Джон Бернерс-Ли**

Любой стандарт W3C проходит 5 стадий согласования:

- Черновик спецификации (*Draft*)
- Рабочий проект (*Working Draft*);
- Последний созыв (*Last Call*);
- Возможная рекомендация (*Candidate Recommendation*);
- Предлагаемая рекомендация (*Proposed Recommendation*);

и только после этого официально становится рекомендацией W3C.



WHATWG (Web Hypertext Application Technology Working Group) — сообщество людей, заинтересованных в развитии Интернета. Было основано в 2004 году производителями браузеров: Apple, Mozilla Foundation и Opera Software. Основным направлением сообщества является развитие HTML и API, необходимого для веб-приложений.

По сути, является подобием **W3C**. WHATWG была недовольна медленными темпами развития стандартов и уклоном W3C в сторону HTML, основанного на XML-синтаксисе

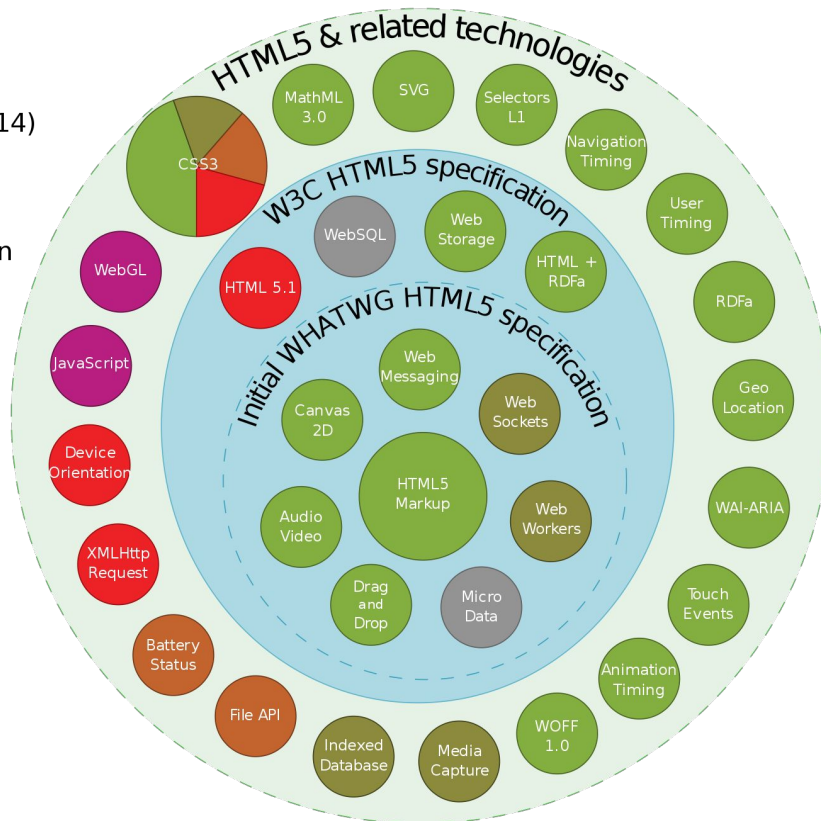
Сейчас WHATWG активно разрабатывает спецификации: HTML, DOM Standard, Fetch Standard, Web workers, Storage Standard, Streams Standard.



HTML5

Taxonomy & Status (October 2014)

- Recommendation/Proposed
- Candidate Recommendation
- Last Call
- Working Draft
- Non-W3C Specifications
- Deprecated or inactive



CSS. Cascading Style Sheets



CSS одна из широкого спектра технологий, одобренных консорциумом W3C и получивших общее название «стандарты Web»

wiki.csswg.org — рабочая группа CSS в рамках консорциума W3C


- CSS level 1 — декабрь 1996
- CSS level 2 — май 1998
- CSS level 2.1 — июнь 2011
- CSS level 3 — с июня 2012 ... (по сути, ещё не принята, постоянно в разработке)
- ~~CSS level 4~~ — никогда не появится, т.к. CSS level 3 разбил все на отдельные модули, которые постоянно развиваются

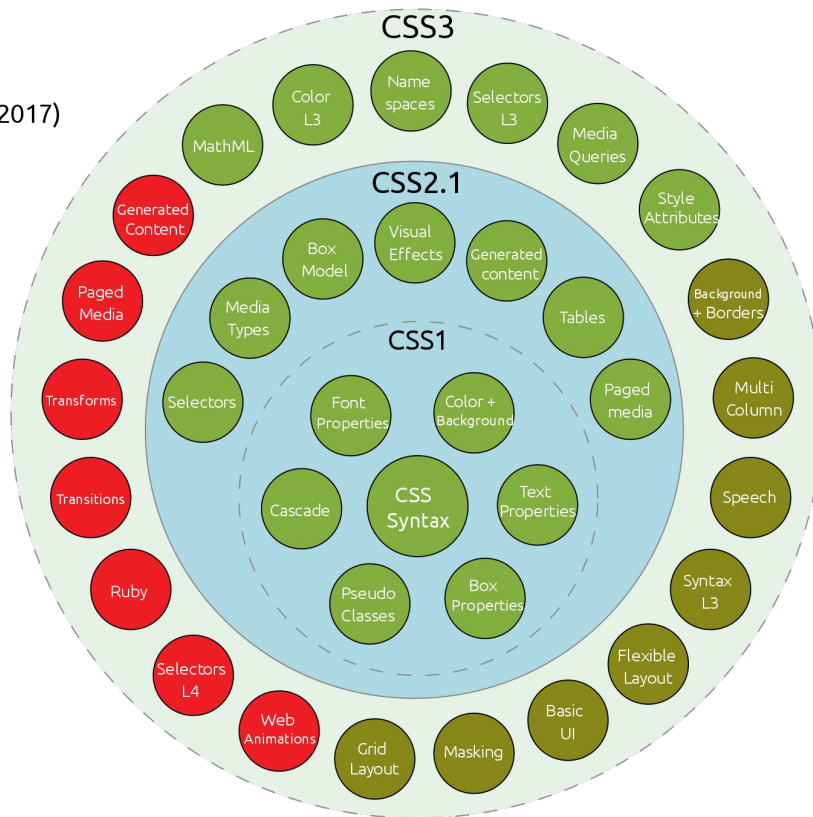
Все CSS-спецификации — www.w3.org/Style/CSS/specs.en.html



CSS3

Taxonomy & Status (September 2017)

-  W3C Recommendation
-  Candidate Recommendation
-  Last Call
-  Working Draft
-  Obsolete or inactive





Возможности

CSS. Управление разметкой. display: block;



Блочные элементы – всегда начинаются с новой строки и занимают всю доступную ширину.

1. `<address>`
2. `<article>`
3. `<aside>`
4. `<blockquote>`
5. `<canvas>`
6. `<dd>`
7. `<div>`
8. `<dl>`
9. `<dt>`
10. `<fieldset>`
11. `<figcaption>`
12. `<figure>`
13. `<footer>`
14. `<form>`
- 15.

1. `<h1>-<h6>`
2. `<header>`
3. `<hr>`
4. ``
5. `<main>`
6. `<nav>`
7. `<noscript>`
8. ``
9. `<p>`
10. `<pre>`
11. `<section>`
12. `<table>`
13. `<tfoot>`
14. ``
15. `<video>`
- 16.

CSS. Управление разметкой. display: inline;



Строчные элементы – всегда **НЕ** начинаются с новой строки и занимают столько ширины, сколько требуется.

1. `<a>`
2. `<abbr>`
3. `<acronym>`
4. ``
5. `<bdo>`
6. `<big>`
7. `
`
8. `<button>`
9. `<cite>`
10. `<code>`
11. `<dfn>`
12. ``
13. `<i>`
14. ``
15. `<input>`
16. `<kbd>`

1. `<label>`
2. `<map>`
3. `<object>`
4. `<output>`
5. `<q>`
6. `<samp>`
7. `<script>`
8. `<select>`
9. `<small>`
10. ``
11. ``
12. `<sub>`
13. `<sup>`
14. `<textarea>`
15. `<time>`
16. `<tt>`
17. `<var>`

CSS. Управление разметкой. Позиционирование элементов



```
1. .container-flex {  
2.     display: flex;  
3. }  
4.  
5. .container-grid {  
6.     display: grid;  
7. }  
8.  
9. .container-center-old {  
10.     display: block;  
11.     width: 150px;  
12.     margin: 0 auto;  
13. }  
14.  
15. .element-left { float: left; }
```

```
1. .container-center-new {  
2.     display: flex;  
3.     justify-content: center;  
4. }  
5.
```

[A Complete Guide to Flexbox](#)

[A Complete Guide to Grid](#)

CSS. Filter

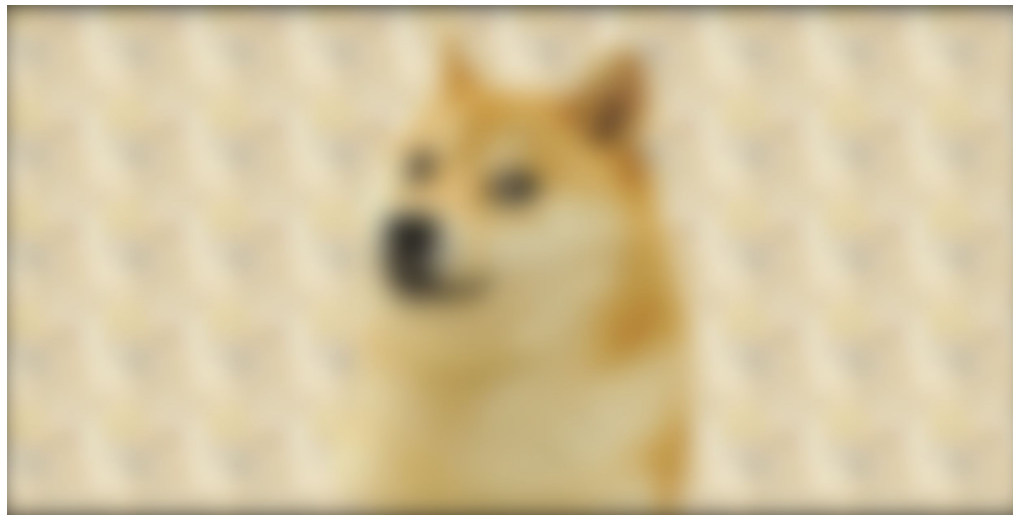


[Пример MDN](#)

[Пример 2](#)

[Пример 3](#)

```
1.  img {  
2.    filter: blur(25px);  
3.  }  
4.
```





Пример MDN

```
1.  /* Apply to 1 property */
2.  /* property name | duration */
3.  transition: margin-right 4s;
4.
5.  /* property name | duration | delay */
6.  transition: margin-right 4s 1s;
7.
8.  /* property name | duration | timing function */
9.  transition: margin-right 4s ease-in-out;
10.
11. /* property name | duration | timing function | delay */
12. transition: margin-right 4s ease-in-out 1s;
13.
14. /* Apply to 2 properties */
15. transition: margin-right 4s, color 1s;
16.
17. /* Apply to all changed properties */
18. transition: all 0.5s ease-out;
19.
```

CSS. Animation



[Пример MDN](#)

[Пример css-tricks](#)

CSS. Transform



[Пример 1](#)

[Пример w3cschools](#)

[Свойство will-change](#)

[Статья про рендер CSS на GPU](#)

[Статья про рендер CSS на GPU 2](#)

CSS. Variables (custom properties)



```
1.  :root {  
2.      --my-color: #FFF;  
3.  }  
4.  
5.  .color-picker {  
6.      box-shadow: 0 0 5em var(--my-color, white);  
7.  }  
8.
```

[Пример MDN](#)

Статья [Medium](#)

CSS. Variables from JS



```
1. // get value
2. document.documentElement.style
3.     .getPropertyValue('--my-color');
4.
5. // set value
6. document.documentElement.style
7.     .setProperty('--my-color', 'green');
8. document.documentElement.style
9.     .setProperty('--my-color', 'var(--fancy-color)');
10.
11.
```

CSS. Variables



Can I use

css variables

? Settings

2 results found

#

CSS Variables (Custom Properties) - CR

Usage
Global

% of all users
90.54% + 0.06% = 90.6%

Permits the declaration and usage of cascading variables in stylesheets.

Current aligned

Usage relative

Date relative

Apply filters

Show all

?

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	Samsung Internet
	12-14		4-47		10-34										
	15	2-30	48	3.1-9	35	3.2-9.2									4
6-10	16-17	31-65	49-72	9.1-11.1	36-57	9.3-11.4		2.1-4.4.4	7	12-12.1			10		5-7
11	18	66	73	12	58	12.1	all	67	10	46	71	64	11	11.8	8.2
		67-68	74-76	12.1-TP		12.2									

Notes

Known issues (3)

Resources (8)

Feedback

1

Enabled through the "Experimental Web Platform features" flag in `chrome://flags`

2

Partial support is due to bugs present (see known issues)



Препроцессоры, инструменты

CSS. Preprocessors



SASS/SCSS — Syntactically Awesome Stylesheets sass-scss.ru

LESS — lesscss.org

Stylus — stylus-lang.com

[Статья](#) про препроцессоры.



```
1.  $font-stack:    Helvetica, sans-serif
2.  $primary-color: #333
3.
4.  .login-form {
5.      color: $primary-color;
6.      &__input {
7.          font: 18px $font-stack;
8.          &:active {
9.              background-color: white;
10.         }
11.     }
12. }
13.
```



```
1.  @mixin border-radius($radius) {  
2.      -webkit-border-radius: $radius;  
3.      -moz-border-radius: $radius;  
4.      -ms-border-radius: $radius;  
5.          border-radius: $radius;  
6.  }  
7.  
8.  .button {  
9.      @include border-radius(10px);  
10. }  
11.
```

CSS. PostCSS



[PostCSS](#) is a tool for transforming styles with JS plugins. These plugins can *lint* your CSS, *support variables and mixins*, *transpile future CSS syntax*, *inline images*, and *more*.

CSS. PostCSS - config



```
1.  // postcss.config.js
2.  module.exports = {
3.    plugins: [
4.      require('precss')({/* ...options */}),
5.      require('autoprefixer')({/* ...options */}),
6.      ...
7.    ]
8.  };
9.
10. // main.js
11. import 'styles.css';
12.
```

CSS. CSS Modules



CSS Modules

```
1.  /* button.css */
2.  .button {
3.      width: 200px;
4.      height: 48px;
5.      border-radius: 12px;
6.  }
7.
8.  .primary {
9.      background-color: green;
10.     font-weight: 500;
11.  }
12.
```

CSS. CSS Modules



```
1.  // button.js
2.  import styles from './button.css';
3.
4.  export default const Button = ({title, primary}) => {
5.      return (
6.          <button
7.              className={`${styles.button} ${primary ?
8.                  styles.primary : ''}`}
9.              >
10.                 {title}
11.            </button>
12.          );
13.  }
```



```
1. <!-- результирующий HTML -->
2.
3. <button class="button-213ge1hw primary-jh4gd318">
4.     Kek!
5. </button>
6. <button class="button-213ge1hw">
7.     Kek is not Schreck
8. </button>
9.
```




JSS

```
1.  // main.js
2.  import jss from 'jss';
3.  import preset from 'jss-preset-default';
4.  import color from 'color';
5.
6.  // One time setup with default plugins and settings
7.  jss.setup(preset());
8.  const styles = {
9.    button: {
10.      width: 200,
11.      background: color('blue').darken(0.3).hex(),
12.    },
13.  };
14.
```



JSS

```
1. // app.js
2. const { classes } = jss.createStyleSheet(styles).attach();
3.
4. document.body.innerHTML = `
5.     <button class="${classes.button}">
6.         Button
7.     </button>
8. `;
9.
```

CSS. Styled Components



Styled Components

```
1.  const Button = styled.a`
2.    display: inline-block;
3.    border-radius: 3px;
4.    padding: 0.5rem 0;
5.    margin: 0.5rem 1rem;
6.    width: 11rem;
7.    background: transparent;
8.    color: white;
9.    border: 2px solid white;
10.
11.    ${props => props.primary && css`
12.      background: white;
13.      color: palevioletred;
14.    `}
15.  `;
16.
```

CSS. Styled Components



Styled Components

```
1.  render(  
2.    <div>  
3.      <Button  
4.    href="https://github.com/styled-components/styled-components"  
5.      target="_blank"  
6.      rel="noopener"  
7.      primary  
8.    >  
9.      GitHub  
10.    </Button>  
11.  
12.    <Button as={Link} href="/docs" prefetch>  
13.      Documentation  
14.    </Button>  
15.  </div>  
16. )  
17.
```



Вопросы?





“

Перерыв! (10 минут)

Препоd (с)



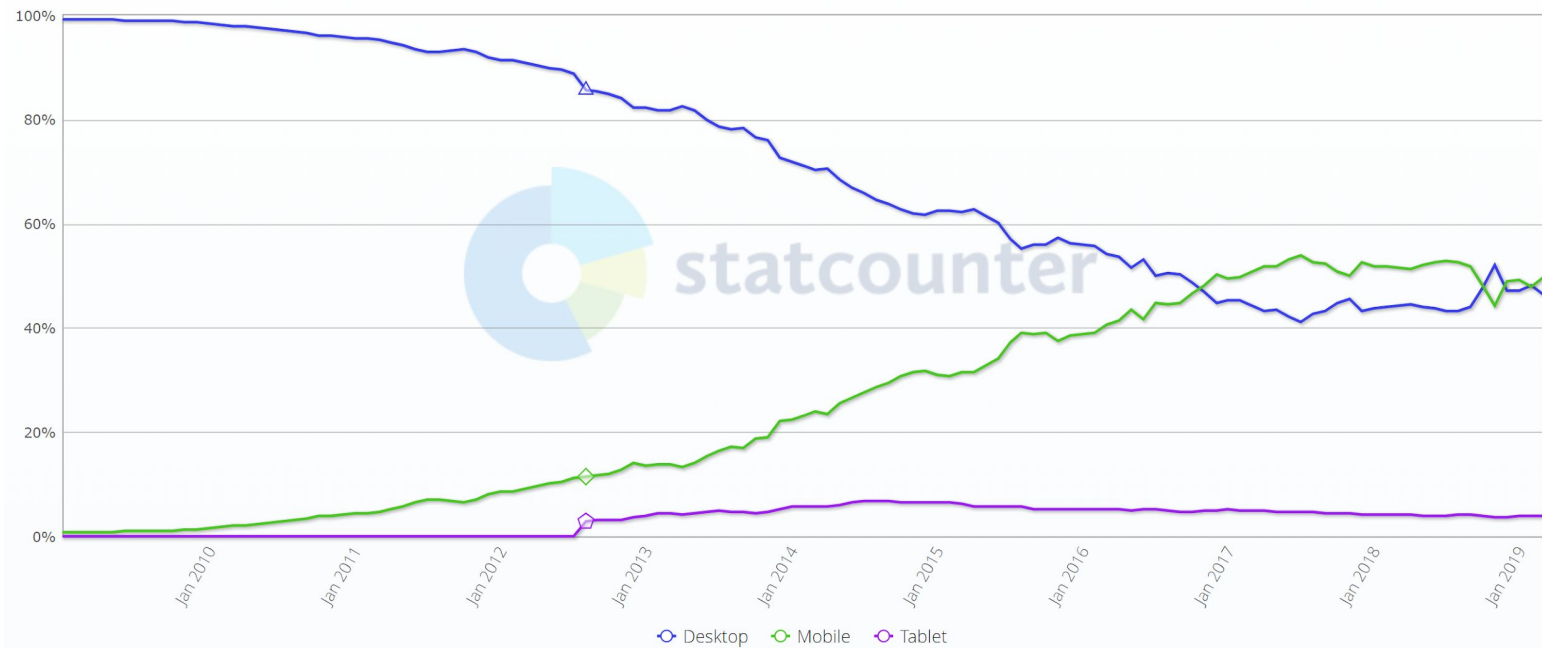
**Разработка под
мобильные устройства**

CSS. Device usage



Desktop vs Mobile vs Tablet Market Share Worldwide

Jan 2009 - Mar 2019



CSS. Native vs. Web stats



Статистика: 1, 6, 12, 13, 14, 17, 19!, 21!

Статистика 2



CSS. Подходы к разработке под мобильные устройства



- Отдельная мобильная версия
- Резиновый (liquid) дизайн
- Отзывчивый или адаптивный (responsive) дизайн
- Mobile-first подход
- **Прогрессивные веб-приложения**
 - graceful degradation
 - progressive enhancement

CSS. Как определить возможности устройства?



- Какой тип устройства (экран, телевизор, часы, микроволновка, тостер...)?
- Какая плотность пикселей?
- Какое разрешение у дисплея
- Какой размер устройства?
- многое другое

CSS. Медиа-запросы (Media Queries)



*A **media query** is a method of testing certain aspects of the user agent or device that the document is being displayed in*

```
1. @media screen and (max-width: 1900px) {  
2.     .container {  
3.         width: 70vw;  
4.         max-width: 1200px;  
5.     }  
6. }  
7.
```

CSS. Медиа-запросы. Использование в CSS



```
1. @media screen and (color) { /* Для цветных экранов */
2.     body { background: whitesmoke; }
3. }
4.
5. /* Для широкоформатных экранов */
6. @media screen and (min-device-aspect-ratio: 16/10) {
7.     ...
8. }
9.
```

CSS. Медиа-запросы. Использование в CSS



```
1.  /* Для широкоформатных экранов */
2.  @media screen and (min-device-aspect-ratio: 16/10) {
3.      ...
4.  }
5.  @media projection { /* проектор */ }
6.  @media handheld { /* смартфоны и носимые устройства */ }
7.  @media tv { /* телевизоры */ }
8.  @media braille { /* устройства для слабовидящих */ }
9.
10. @media (hover) { /* если на устройстве работает hover */ }
11.
12. @media (pointer: coarse) { /* тачскрины, управляемые пальцами
    */ }
13. @media (pointer: fine) { /* мышь или стилус */ }
14. @media (pointer: none) { /* нет курсора */ }
15.
```

CSS. Медиа-запросы. Использование в HTML



```
1. <link rel="stylesheet"
2.     media="all and (orientation : portrait)"
3.     href="portrait.css">
4.
5. <link rel="stylesheet"
6.     media="all and (orientation : landscape)"
7.     href="landscape.css">
8.
9.
```

CSS. CSS директива @supports



```
1. @supports (display: flex) {  
2.     div { display: flex; }  
3. }  
4.  
5. @supports not (display: flex) {  
6.     div { float: left; } /* задан альтернативный стиль */  
7. }  
8.
```


CSS. CSS директива @supports



```
1.  @supports (display: -webkit-flex) or
2.      (display: -moz-flex) or
3.      (display: flex) {
4.
5.      /* добавляем сюда ваших клёвых стилей */
6.  }
7.
8.  // использование в JS
9.  const supportsFlex1 = CSS.supports('display', 'flex');
10. const supportsFlex2 = CSS.supports('(display: flex)');
11.
```



Единицы измерения в CSS



Спецификация

Относительные

- `em` — font size of the element
- `ex` — x-height of the element's font
- `ch` — width of the "0" (ZERO, U+0030) glyph in the element's font
- `rem` — font size of the root element
- `vw` — 1% of viewport's width
- `vh` — 1% of viewport's height
- `vmin` — 1% of viewport's smaller dimension
- `vmax` — 1% of viewport's larger dimension



Спецификация

Абсолютные

- **cm** — $1\text{cm} = 96\text{px}/2.54$
- **mm** — $1\text{mm} = 1/10\text{th of } 1\text{cm}$
- **Q** — $1\text{q} = 1/40\text{th of } 1\text{cm}$
- **in** — $1\text{in} = 2.54\text{cm} = 96\text{px}$
- **pc** — $1\text{pc} = 1/6\text{th of } 1\text{in}$
- **pt** — $1\text{pt} = 1/72\text{th of } 1\text{in}$
- **px** — $1\text{px} = 1/96\text{th of } 1\text{in}^*$



[Статья](#)

[Статья 2](#)

- **Аппаратные пиксели** — физический пиксель матрицы дисплея
- **Аппаратно-независимые пиксели** (Device-independent pixels, dip) — пиксели дисплея, приведённые к единому масштабу, чтобы соответствовать примерно одинаковому углу зрения на всех девайсах (с учётом расстояния, на котором мы их держим)
- **Пиксель CSS** — единица измерения вёрстки

CSS. Retina





[Статья Retina](#)

RETINA

2880px across

1800px high

HTML
OBJECT

physically
800x600
on screen

Actual CSS

```
div {  
  width: 400px;  
  height: 300px;  
}
```

NORMAL

1440px across

900px high

HTML
OBJECT

physically
400x300
on screen

Actual CSS

```
div {  
  width: 400px;  
  height: 300px;  
}
```



**Область просмотра
(viewport)**

CSS. Область просмотра (viewport)



CSS. Область просмотра (viewport)



1. `<meta name=viewport`
2. `content="width=device-width, initial-scale=1">`
- 3.

Значения:

- width, height
- initial-scale
- minimum-scale, maximum-scale
- user-scalable

CSS. Область просмотра (viewport)



```
1.  @viewport {  
2.      width: device-width;  
3.      height: device-height;  
4.      zoom: 2;  
5.      user-zoom: fixed;  
6.  }  
7.
```



Нативное взаимодействие

CSS. Нативное взаимодействие



Проблемы:

- Зоопарк событий
- 300-ms задержка
- Нет нужных событий



События mouse-events:

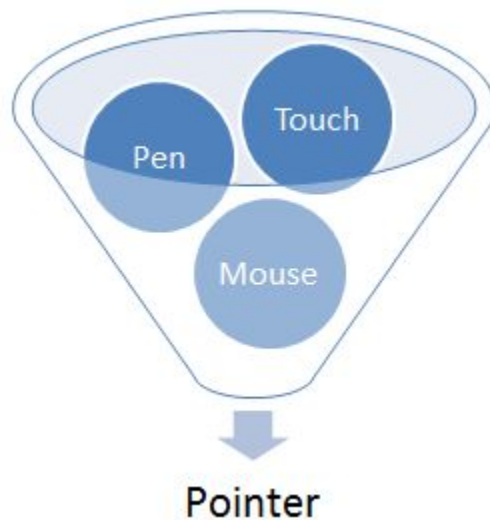
- mousedown
- mouseup
- click
- dblclick
- contextmenu
- mouseover (-out, -move)

CSS. Зоопарк событий



События touch-events:

- touchstart
- touchmove
- touchcancel
- touchend





События pointer-events:

- pointerdown
- pointerup
- pointercancel
- pointermove
- pointerover
- pointerout
- gotpointercapture
- lostpointercapture

CSS. Зоопарк событий



Can I use ?  Settings

5 results found

CSS pointer-events (for HTML) - UNOFF

Usage % of all users ?
Global 96.55%

This CSS property, when set to "none" allows elements to not receive hover/click events, instead the event will occur on anything behind it.

Current aligned	Usage relative	Date relative	Apply filters	Show all	?										
IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Opera Mobile	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Balc Brow:
		2-3.5		3.1-3.2	10-12.1										
6-10	12-17	3.6-68	4-76	4-12.1	15-60	3.2-12.3		2.1-4.4.4	12-12.1				4-9.2		
11	18	69	77	13	62	13.1	all	76	46	76	68	12.12	10.1	1.2	7.1
	76	70-71	78-80	TP											

CSS. 300-ms задержка



Между событием touchend и click проходит 300-350ms

```
1.  html {  
2.      touch-action: manipulation;  
3.      touch-action: auto;  
4.      touch-action: pan-x;  
5.      touch-action: pinch-zoom;  
6.  }  
7.  
8.
```

CSS. Дополнительные события



- pan
- swipe
- rotate
- pinch/zoom
- doubletap

CSS. Дополнительные события. Решение



```
1. // HammerJS - http://hammerjs.github.io/  
2. const hammer = new Hammer(element, options);  
3. hammer.on('swipe', function(event) {  
4.     console.log(event);  
5. });  
6.
```

CSS. PWA, TWA



PWA:

<https://developers.google.com/web/progressive-web-apps/checklist>

<https://habr.com/ru/company/google/blog/414609/>

<https://web.archive.org/web/20151103001838/http://www.luster.io/blog/9-29-14-mobile-web-checklist.html>

TWA:

<https://developers.google.com/web/updates/2019/02/using-twa>

<https://habr.com/ru/post/439238/>

[Lighthouse](#)

Домашнее задание № 4



1. Добавить “украшательства”, анимации, транзишны
2. Добавить viewport
3. Приблизить дизайн своего приложения к макетам
4. ...

Срок сдачи

28 октября



Спасибо за внимание!