



# Дати (проект)

вересень 2017						
нд.	пн.	вт.	ср.	чт.	пт.	сб.
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

жовтень 2017						
нд.	пн.	вт.	ср.	чт.	пт.	сб.
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

 Бажано  
 Обов'язково

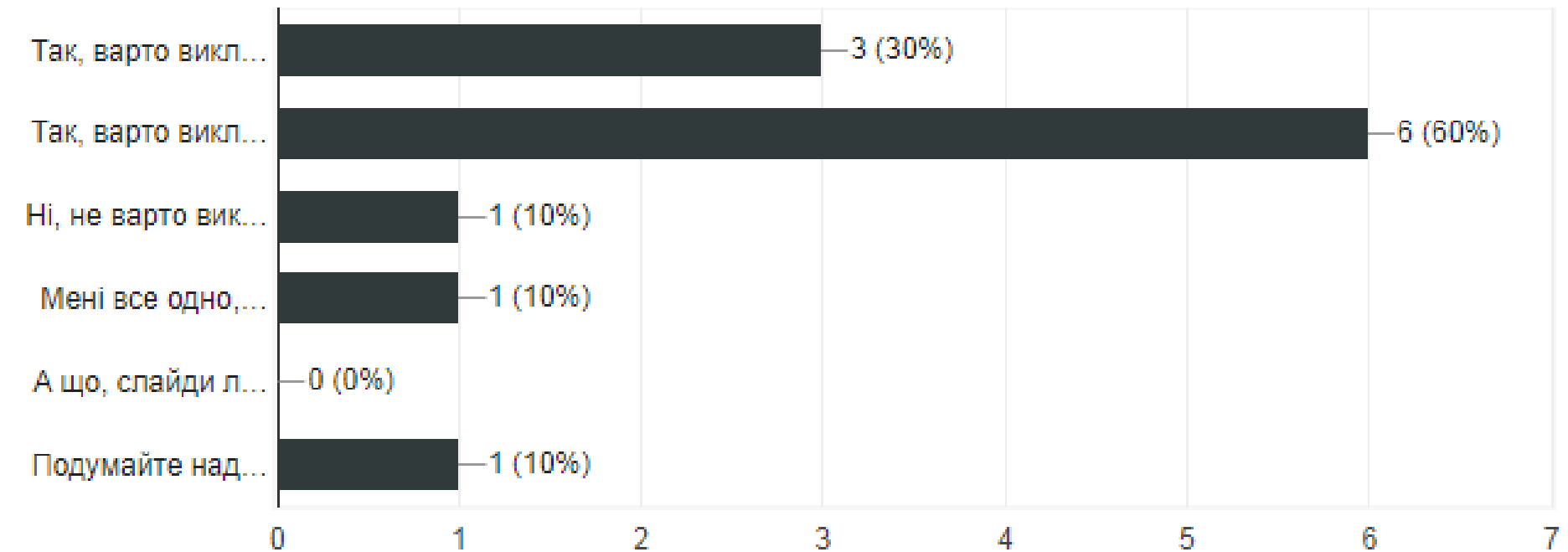
- Дедлайн цього тижня (9 жовтня)
  - **Обов'язково:** вибрати тему проекту, технології, коротко описати вимоги
  - Виконали: ~~13~~ 16 з 21 (~~62~~ 76%)
  - Описали вимоги: ~~9~~ 14 з 21 (~~43~~ 67%)
  - Всі інші хочуть втрачати бали? 😊
- Дедлайн через тиждень (23 жовтня)
  - **Бажано:** перша демонстрація, розгорнутий проект, репозиторій з кодом

# Результати мікро-опитування

- Загальна кількість відповідей: 10 (48%)
  - 3 коментарями/поясненнями: 9
  - Без коментарів: 1
- Всього балів:  $9 * 0,2 + 1 * 0,1 = 1,9$  (45% від максимально можливих)

# Питання 2: Чи варто викладати попередню версію слайдів лекції завчасно, тобто перед лекцією?

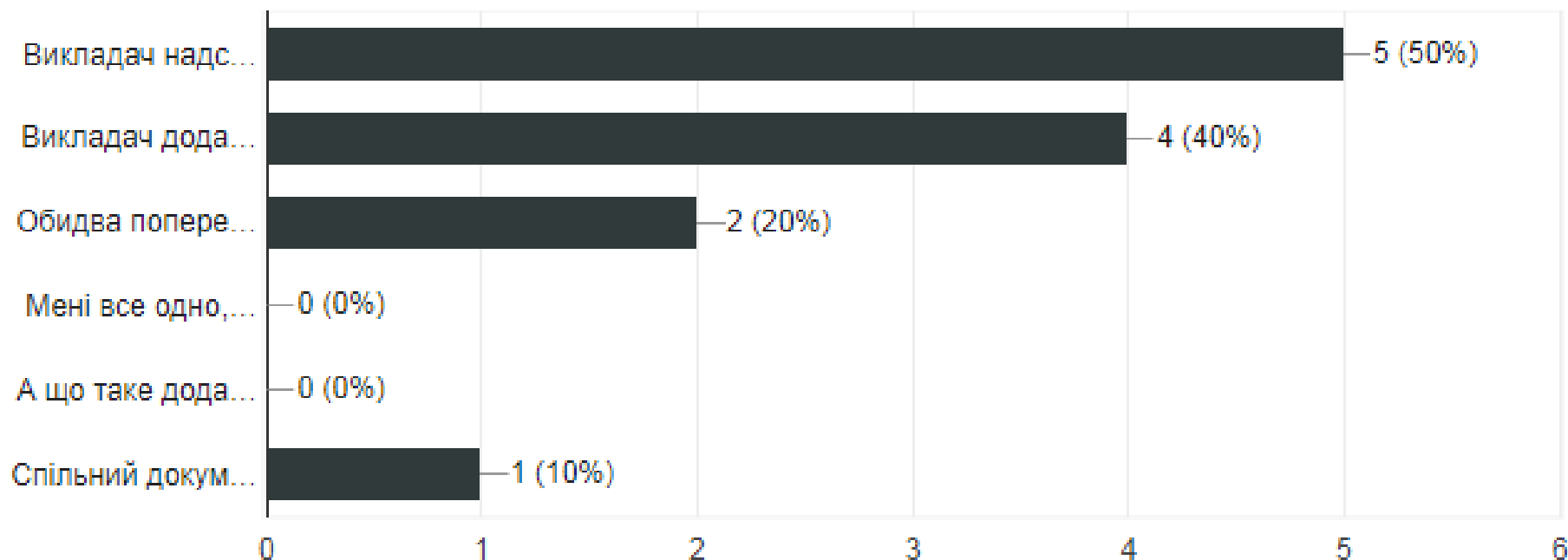
10 responses



- Майже однозначна позиція – треба викладати
- Переважна більшість – перед парою, а не за день

## Питання 1: Як краще надсилати коментарі щодо відповідей на додаткові питання?

10 responses



- Думки розділились, трохи більше за лише e-mail
- Компромісний варіант: на e-mail одразу, в shared document через пару?

# Більше обговорень лекцій та питань?

План наступної лекції завчасно (за 1-2 тижні до пари),  
з можливістю обговорення та питань, які будуть враховані в лекції?  
+

Editable shared document for questions and answers?  
-

GitHub repository for questions and answers?  
+  
issues++

Slack workspace (or alternatives) for questions and answers?  
-

# Додаткові питання

## Загальні коментарі

- Не варто повторюватись
  - Те, що було в слайдах
  - Те, що було у попередніх відповідях інших студентів
- У більшості питань є достатня кількість можливих відповідей, щоб обійтись без повторів
- Інший ресурс, що підтверджує ту саму точку зору – нормально
  - Але інші точки зору – цікавіше
- Список – бажано вказати, що нового та що видається цікавим

# 1.1. Виклики розробки

- <https://wpbusinessstips.com/2015/01/complexity-web-development-simple-guide/>
  - Важко прогнозувати складність проекту
  - Вивчення нових технологій
  - Нові комбінації технологій
  - Різноманіття середовищ
  - Непередбачувані проблеми
- <https://habrahabr.ru/post/15906/>
  - Типові проблеми HTML/CSS верстки
- Типові проблеми з безпекою
  - <https://www.sans.edu/cyber-research/security-laboratory/article/secure-code-sec545>
  - <http://www.creativeblog.com/web-design/website-security-tips-protect-your-site-7122853>
  - <https://www.secureworks.com/capabilities/security-risk-consulting/app-testing/web-app-security>

# 1.2. Історія мереж (до 2000р.)

- 1990 – [Archie](https://inforesist.org/kak-poyavilas-pervaya-poiskovaya-sistema-archie/) (FTP Search)
  - First Internet search engine? (before Web)
  - <https://inforesist.org/kak-poyavilas-pervaya-poiskovaya-sistema-archie/>
- 1998 - Google web search
  - Було на слайдах
  - <https://www.google.com/intl/en/about/our-story/>
  - <http://dnevnyk-uspeha.com/interesnye-fakty/kompanija-google-istorija-sozdaniya.html>
  - <https://copirayter.ru/istoriya-sozdaniya-yandeks-i-gugl/>
  - <http://webtrafff.ru/kratkaya-istoriya-razvitiya-google.html>
  -





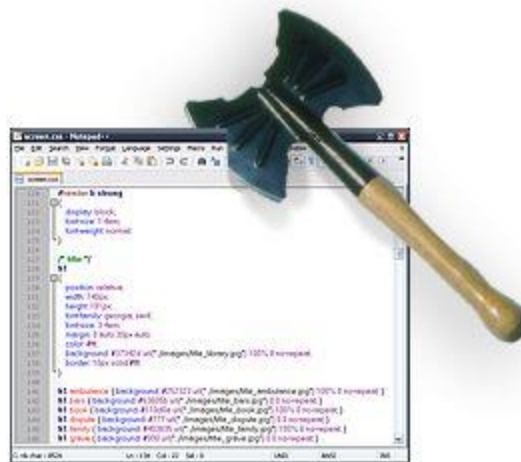
## 2.1. Security (backend)

- <https://www.symantec.com/connect/articles/five-common-web-application-vulnerabilities>
  - Remote Code Execution
  - Format String Vulnerability
  - Username Enumeration (*not very useful?*)
  - SQL Injection, Cross-Site Scripting (XSS) (*було на слайдах*)
- <https://www.toptal.com/security/10-most-common-web-security-vulnerabilities>
  - [https://www.owasp.org/index.php/Top\\_10\\_2013-Top\\_10](https://www.owasp.org/index.php/Top_10_2013-Top_10)
  - Insecure Direct Object Reference



# Лекція 3.

## Розробка клієнтської частини веб-застосунків



# Ви це вже знаєте?

- Мова HTML
  - Теги, елементи, атрибути, [DOCTYPE](#)
  - Інтеграція скриптів, стилів
- Мова CSS
  - Selectors, properties, rules
  - Priority, !important
  - Псевдокласи, псевдоелементи
- Layout: absolute, relative, float, z-index, box model, column, table/grid
- [Mobile](#): viewport, media queries, responsive design
- Мова JavaScript
  - Синтаксис
  - Об'єктно-орієнтовані можливості, прототипи
  - “use strict”
- JavaScript APIs: DOM, AJAX / XMLHttpRequest, LocalStorage, WebSockets, WebWorkers
- Preprocessors: Less, Sass, CoffeeScript, TypeScript
- Бібліотеки: Bootstrap, jQuery, Angular, React
- Інструменти: мініфікатори, unit testing, static analysis, build



# Особливості сучасного frontend development (2017)

костилі +++ shims

забагато бібліотек залежності на 2 екрани

відхід від vanilla JS/CSS

less sass stylus dying out

typescript grows

coffescript dies

ES6+ grows

JS modules

wasm???

accessibility

web components

react must die?

functional programming approach

immutable

# Особливості сучасного frontend development (2016)

все має бути красиво

для тих, хто не вміє програмувати :)

responsive design

user friendly

AJAX (client-side logic in JS)

ads

APIs (google, fb, ...)

cross-browser

lots of frameworks

develop fast

SEO?

less, sass

html5

localStorage

jquery is not the dominating framework

# Особливості сучасного frontend development

- Немає контролю над середовищем виконання
  - Версія браузера, ОС
  - Розмір екрану
- Результати розробки безпосередньо помітні користувачу
- Можна подивитись, як воно працює у інших
- Швидка зміна технологій

# Цікаві нові тенденції в frontend development (2017)

standards

ES hosted on github

# Цікаві нові тенденції в frontend development (2016)

angular 2

bootstrap 3? 4?

ES7

CSS3

React 15

material design

node.js (backend)

webpack, gulp

reduct, flux

single page app



# Клієнтська частина

- HTML
- CSS
- JavaScript
  - Core language
  - APIs (DOM, AJAX, canvas, local storage, web sockets, web workers, ...)
- Plug-ins (Flash, Silverlight, ...) – *finally dead?*
- Інші формати (SVG, MathML, XForms, ...)

# Frontend Developer = 2 Meanings

- 1. Frontend = HTML + CSS + JavaScript (a little)
  - “Frontend Designer”
- 2. Frontend = JavaScript + frameworks + toolchain
  - “Frontend Developer”
  - Спеціалізація: Angular Developer, React Developer, ...

# Мова HTML

- Елементи (теги)
  - Текст `<span>Hello world</span> <div>`
  - Зображення ``
  - Посилання `<a href="/page.php">`
  - Таблиці `<table> <tr> <td>`
  - Списки `<ol> <ul> <li>`
  - Форми `<form> <input> <button>`
  - Зовнішні об'єкти `<object>`
  - Скрипти, стилі `<script> <style> <link>`
- Атрибути `<a href="/page.php">`
- Події `<a onclick="openWindow(123);return false;">`
- Block та inline елементи



Document Outline		Lists		Objects	
<!DOCTYPE>	Version of (X)HTML	<ol>	Ordered list	<object>	Object
<html>	HTML document	<ul>	Unordered list	<param />	Parameter
<head>	Page information	<li>	List item		
<body>	Page contents	<dl>	Definition list		
		<dt>	Definition term		
		<dd>	Term description		
Comments		Forms		Empty Elements	
<!-- Comment Text -->		<form>	Form	<area />	<img />
		<fieldset>	Collection of fields	<base />	<input />
		<legend>	Form legend	 	<link />
		<label>	Input label	<col />	<meta />
		<input />	Form input	<hr />	<param />
		<select>	Drop-down box		
		<optgroup>	Group of options		
		<option>	Drop-down options		
		<textarea>	Large text input		
		<button>	Button		
Page Information		Tables		Core Attributes	
<base />	Base URL	<table>	Table	class	style
<meta />	Meta data	<caption>	Caption	id	title
<title>	Title	<thead>	Table header	Note: Core Attributes may not be used in base, head, html, meta, param, script, style or title elements.	
<link />	Relevant resource	<tbody>	Table body		
<style>	Style resource				
<script>	Script resource				
Document Structure				Language Attributes	
<h[1-6]>	Heading			dir	lang
<div>	Page section			Note: Language Attributes may not be used in base, br, frame, frameset, hr, iframe, param or script elements.	
<span>	Inline section				
<p>	Paragraph				
 	Line break				
<hr />	Horizontal rule				

# Мова CSS

- Стилi
- Прив'язування до елементiв (селектори)
  - Тег `div`
  - Клас `div.mainText`
  - ID `#headerLogo`
- Властивостi `.mainText {font-size:12px;}`
- Псевдокласи: `:link, :visited, :hover, :active`
- `!important`



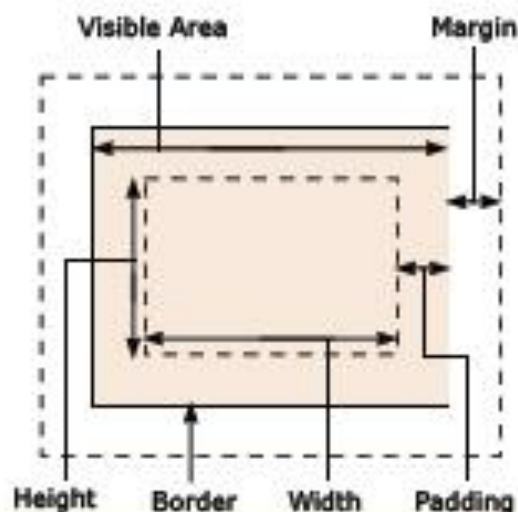
## Selectors

*	All elements
div	<div>
div *	All elements within <div>
div span	<span> within <div>
div, span	<div> and <span>
div > span	<span> with parent <div>
div + span	<span> preceded by <div>
.class	Elements of class "class"
div.class	<div> of class "class"
#itemid	Element with id "itemid"
div#itemid	<div> with id "itemid"
a[attr]	<a> with attribute "attr"
a[attr='x']	<a> when "attr" is "x"
a[class~='x']	<a> when class is a list containing 'x'
a[lang]='en']	<a> when lang begins "en"

## Pseudo-Selectors and Pseudo-Classes

:first-child	First child element
:first-line	First line of element
:first-letter	First letter of element
:hover	Element with mouse over
:active	Active element
:focus	Element with focus
:link	Unvisited links

## Box Model



## Positioning

display	clear
position	z-index
top	direction +
right	unicode-bidi
bottom	overflow
left	clip
float	visibility

## Dimensions

## Boxes

margin x	border-color x
margin-top	border-top-color
margin-right	border-right-color
margin-bottom	border-bottom-color
margin-left	border-left-color
padding x	border-style x
padding-top	border-top-style
padding-right	border-right-style
padding-bottom	border-bottom-style
padding-left	border-left-style
border x	border-width x
border-top x	border-top-width
border-bottom x	border-right-width
border-right x	border-bottom-width
border-left x	border-left-width

## Tables

caption-side +	border-spacing +
table-layout	empty-cells +
border-collapse +	speak-header +

## Paging

size	page-break-inside +
marks	page +

# HTML і CSS

- HTML
  - Мова розмітки
  - Описує структуру документа
- CSS
  - Мова стилів
  - Описує зовнішній вигляд документа
- Розділення структури і стилів
  - Теги відповідають структурі документа
  - Стили не описуються в самих тегах
  - Класи відповідають призначенню, а не зовнішньому вигляду (confirm-button замість green-button)



HTML & CSS

# Розташування елементів за допомогою стилів



- Tableless web layout
- Використовувати для розташування елементів властивості стилів
  - margin, padding, ...
  - float
- Не використовувати таблиці (table, tr, td) для розташування елементів, що не є таблицями

## <TABLE TAGS>

```
<table cellpadding="0" cellspacing="0" border="0">
<tr>
<td colspan="3" height="120px">...</td>
</tr>
<tr>
<td class="nav" valign="top">...</td>
<td class="content" valign="top">...</td>
<td class="left" valign="top">...</td>
</tr>
<tr>
<td colspan="3">...</td>
</tr>
</table>
```

## <DIV TAGS>

```
<div id="header">...</div>
<div id="nav">...</div>
<div id="content">...</div>
<div id="left">...</div>
<div id="footer">...</div>
```

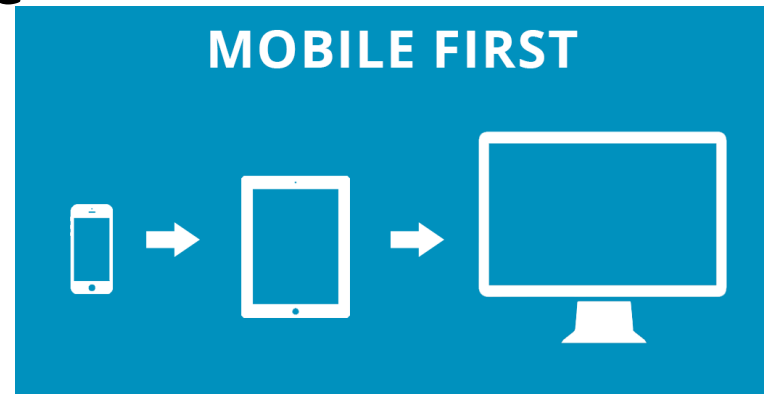


# Переваги

- Менший розмір сторінки
- Простіше вносити зміни
- Однорідність зовнішнього вигляду
- Підтримка нестандартних пристроїв

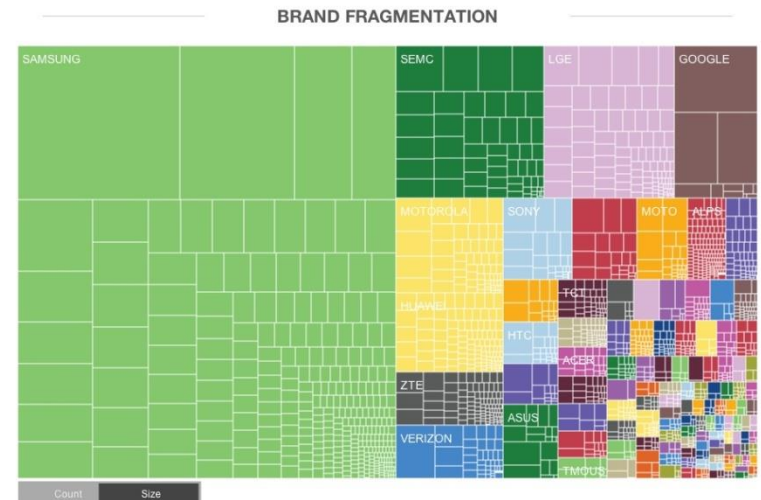
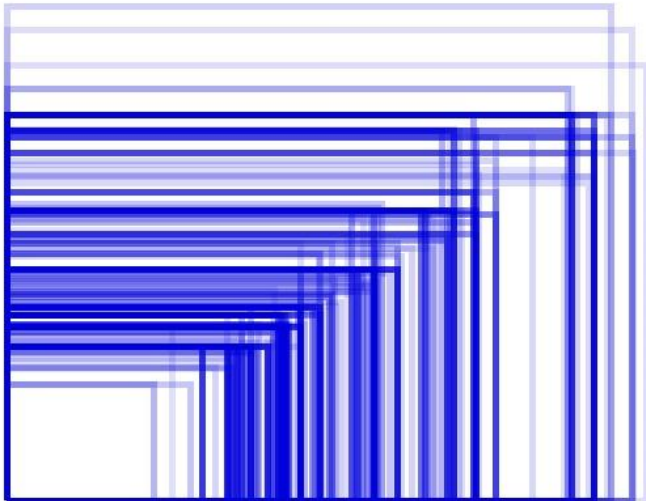
# Мобільні сайти

- Mobile First
  - Спочатку розробляємо версію, що працює на всіх пристроях
  - Потім додаємо додаткові можливості для більш потужних пристроїв
  - А не створюємо лише для desktop, а потім намагаємось додати mobile

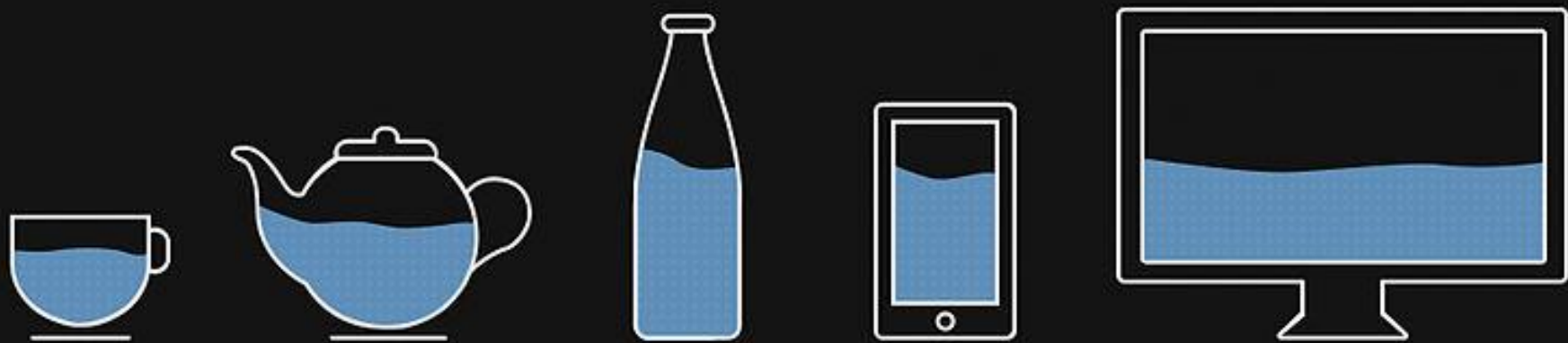


# Responsive Web Design

- Можна робити окрему версію сайту для кожного пристрою
  - Але пристроїв зараз дуже багато...
- Краще реалізувати одну версію, яка буде працювати всюди



# CONTENT IS LIKE WATER



“ You put water into a cup it becomes the cup.  
You put water into a bottle it becomes the bottle.  
You put it in a teapot, it becomes the teapot. ”

# Як реалізовувати responsive web design? (2017)

viewport  
mediaqueries  
useragent

lazy images  
<picture>  
svg

# Як реалізовувати responsive web design?

many IFs  
bootstrap  
media query  
hire CSS guru ninja :)

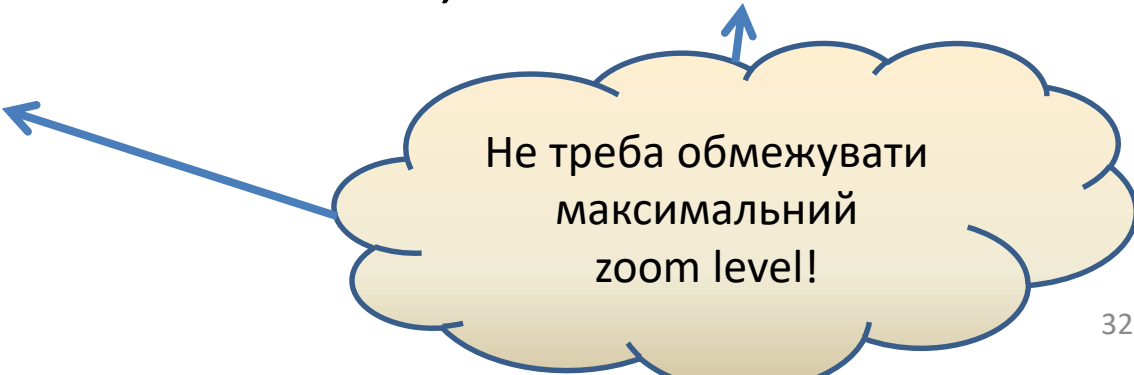
# Як реалізовувати responsive web design?

- viewport
- Responsive grid
- Media queries
- Fluid images



# Viewport

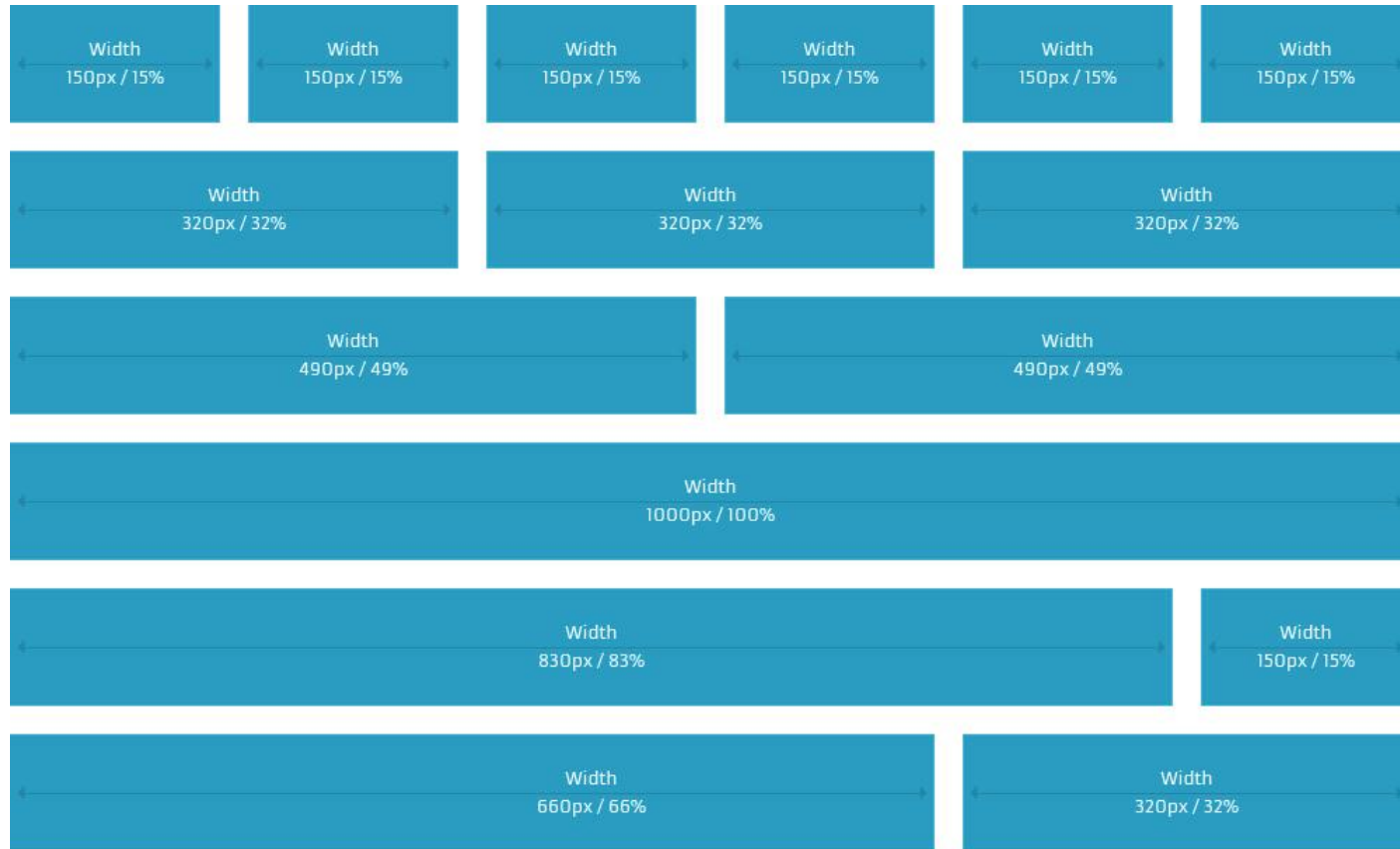
- `<meta name="viewport" content="width=device-width, initial-scale=1">`
- Не є частиною стандартів
  - Спочатку Apple, потім інші мобільні браузерери
- Властивості
  - width, height (not supported)
  - initial-scale, minimum-scale, maximum-scale
  - user-scalable



Не треба обмежувати  
максимальний  
zoom level!



# Responsive grid



- Розраховано на певну кількість колонок
- Ширина не в абсолютних одиницях, а у % відносно контейнера
- При зменшенні ширини екрану – всі елементи в одну колонку

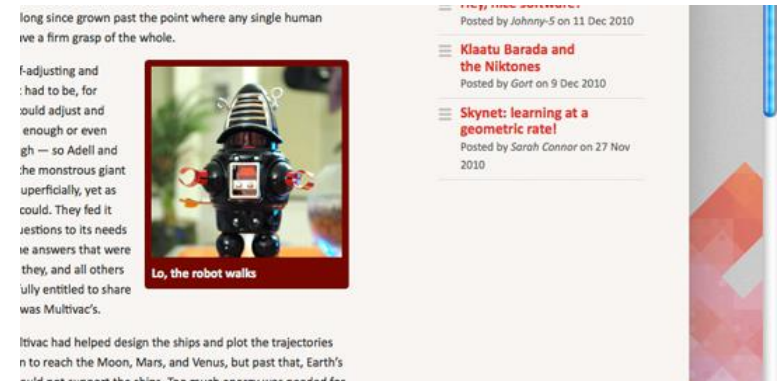
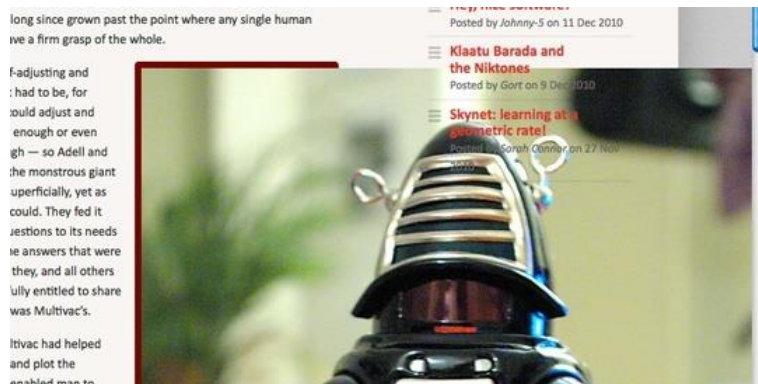
# Media Queries

- `<link rel="stylesheet" media="screen and (min-width: 700px)" href="style.css">`
- `@media orientation: landscape { ...}`
- Дозволяють застосовувати окремі правила лише для частини пристроїв
- Не варто зловживати
  - Краще одне універсальне правило
  - Але якщо не виходить – краще media queries, ніж якісь CSS hacks



# Fluid images

- `img { max-width: 100%; }`
- Ширина зображення обмежена шириною контейнера
  - Великий контейнер – зображення повного розміру
  - Маленький контейнер – зменшене зображення



# Twitter Bootstrap

- Mobile first
- Responsive grid
- Components, icons
- Plugins



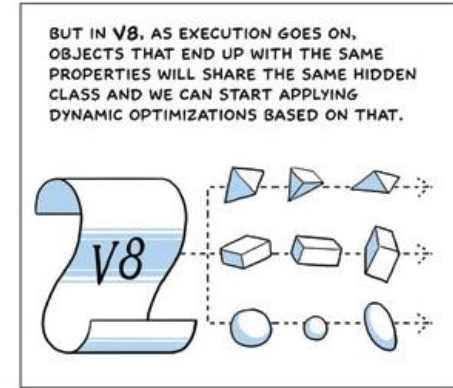
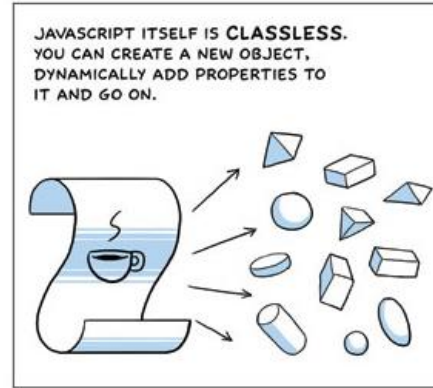
# CSS preprocessors

- Найбільш популярні: Sass, Less
- Зменшують повтори коду
  - Nesting
  - Selector Inheritance
  - Mixins
- Наближують до мов програмування
  - Variables
  - Functions
  - Loops
- Компілюються в CSS



# Мова JavaScript

- Скриптова мова
- Динамічна типізація
- Об'єктно-орієнтована
  - Прототипи замість класів
- Елементи функціональної мови
- Назва
  - Не плутати з Java!
  - ECMAScript, JScript




# Обмежуйте можливості JavaScript

- Багато старих можливостей
  - Часто приховують помилки
- “use strict”;

```
6 "use strict";
7
8 function calculate() {
9     abc = 42;
10
11     // go get the subtotal and tip amounts from the page
12     var subtotalBox = document.getElementById("subtotal");
13     var tipBox = document.getElementById("tip");
```

✖ Uncaught ReferenceError: abc is not defined

# Інтеграція в HTML

- Елемент **script**
  - Inline
  - Окремий файл (.js) – найкращий варіант
- Обробники подій (**onload, onfocus, ...**)
- Схема **javascript:** 
  - Обмеження запуску з address bar





# Frontend Toolchain (2017)

webpack  
gulp grunt  
brunch

postcss  
normalize  
autoprefixer  
uglify  
cssnano

webpack plugin  
unit tests

phantomjs -> headless chrome  
npm yarn bower

babel

# Frontend Toolchain (2016)

gulp webpack

babel

npm

preprocessor (less sass coffeescript)

minifier, obfuscator

# Frontend Toolchain



- (Node.js package manager – [npm](#) )
- Task runner / build system – [Grunt](#), [Gulp](#)
- Package manager – [Bower](#)
- Scaffolding – [Yeoman](#)
- Module loader – [CommonJS](#) / [AMD](#),  
[RequireJS](#), [Browserify](#), [webpack](#)



YEOMAN

# Automatic tasks

- Пошук помилок (статичний аналіз): [JSLint](#), [JSHint](#), [ESLint](#), [Flow](#), [Plato](#)
- Unit testing: [QUnit](#), [Jasmine](#), [Mocha](#), [Karma](#), [Sinon.JS](#)
- UI testing: [Selenium](#), [CasperJS](#), [Protractor](#)
- Документація: [JSDoc](#), [Docco](#), [YUIDoc](#), [JSDuck](#)
- Зменшення розміру: [Closure Compiler](#), [UglifyJS](#), [JSBeautifier](#)
  - Closure Compiler != [Clojure Language](#), ClojureScript



# Інші інструменти

- IDE: WebStorm, Aptana Studio, Eclipse JSDT, NetBeans, Visual Studio/Visual Web Developer, Komodo IDE
- Debug: Developer Tools (in browser: Opera Dragonfly, WebInspector, ...), Firebug, *Venkman*, *CompanionJS*
- Розширення мови: [Babel](#) (ES6+), [CoffeeScript](#), [TypeScript](#)
- Генерація JavaScript з іншої мови: [Google Web Toolkit](#), [Scala.js](#), [pyjs](#), [Script#](#), [Emscripten](#) / [Asm.js](#)
- Source control, bug tracker, ...

# Obsolete

- Static analysis: [JSAnalyse](#) , [JSure](#)
- Тестування: [JSUnit](#), [J3Unit](#), [YUI Test](#),  
*JSTestDriver*, *FireUnit*, *jspec*, *JSLitmus*
- Документація: [jGrouseDoc](#)
- Зменшення розміру: [YUI Compressor](#),  
[JavaScript Minifier](#), [JSMIN](#), [Dojo ShrinkSafe](#),  
[Packer](#), [JSCompress](#)
-

# JavaScript Frameworks/Libraries (2017)

react vue  
redux vue

angular

polymer

knockout  
backbone

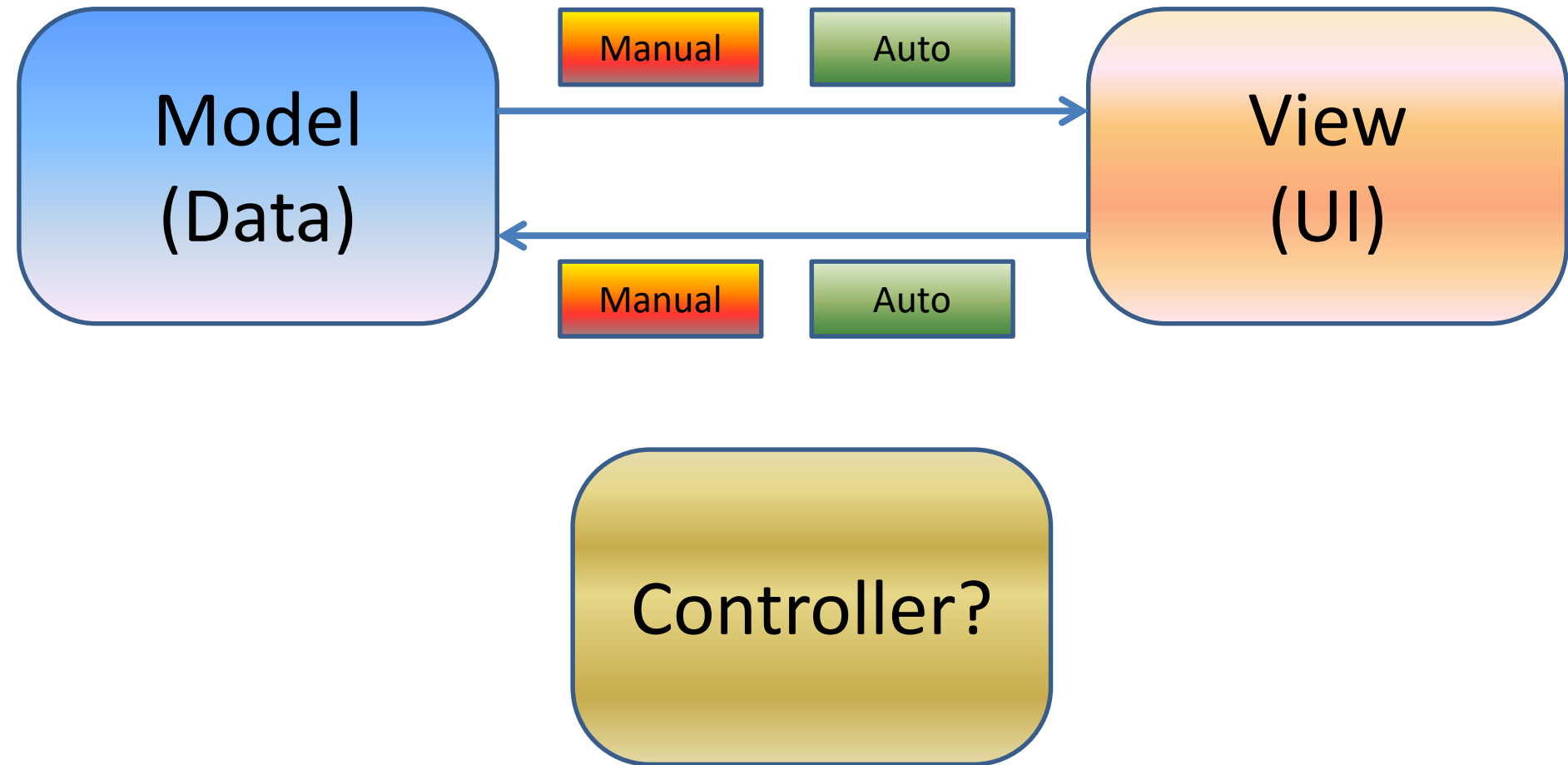
typescript  
clojurescript  
purescript  
elm  
coffeescript

# JavaScript Frameworks/Libraries (2016)

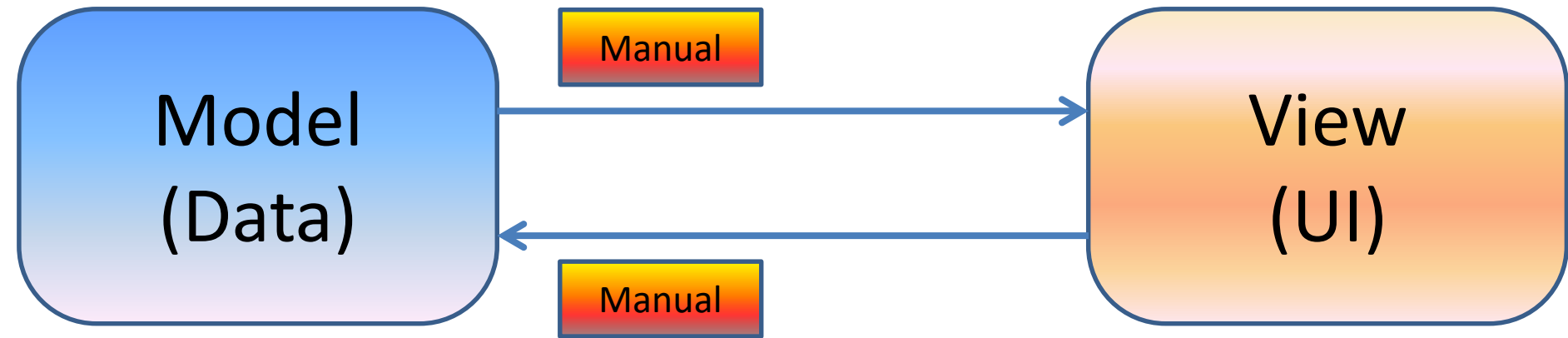
angular  
react`  
backbone  
jquery



# Frontend Frameworks



# 1. Simple frameworks/libraries



# Simple frameworks/libraries

- Базова структура в HTML/CSS
- JavaScript підставляє значення в потрібних місцях
- Обробка подій та зміна моделі
- Готові компоненти
- Популярні приклади
  - [jQuery](#), [YUI](#), [Dojo](#), [MooTools](#), ...



# JavaScript Frameworks

## jQuery

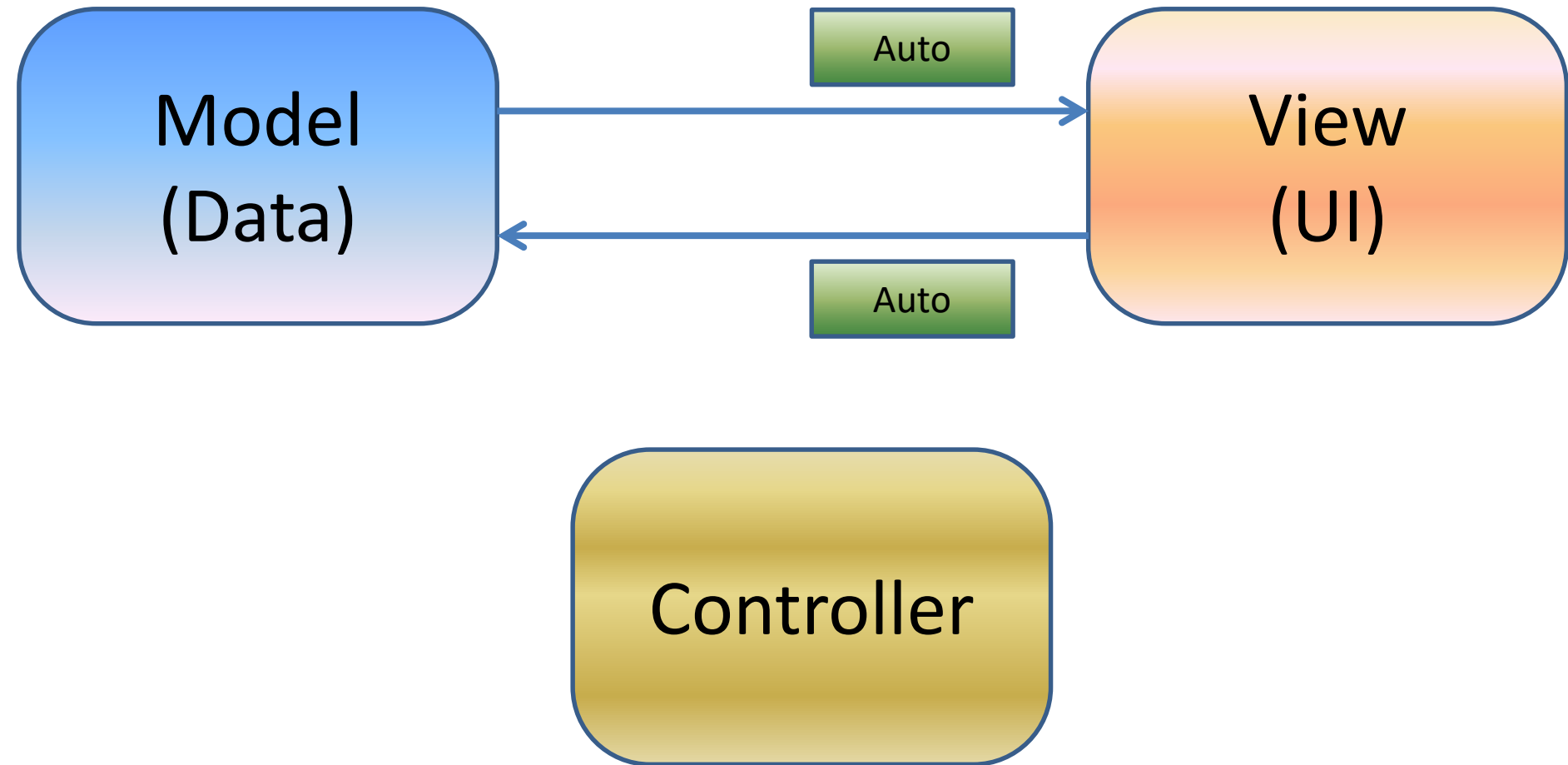
- Спрощений доступ до елементів `$('.myclass')`
- Робота з DOM `$('.a.b').addClass('c'); $('.a').append('<br>');`  
`$('.a').remove(); $('.a').html('');`
- Події `$('.a').click(function() {alert('Clicked!');});`
- Підтримка AJAX `$.post('svc.php', function(data) {`  
`$('.res').html(data); }); $.ajax(...)`
- Елементи керування `$('#lst').autocomplete({source:getLst});`
- Графіка, ефекти і анімація `$('.a').animate({opacity:0.5},500);`  
`$('.a').fadeTo(500,0.5);`
- Службові функції `$.isArray(); $.merge(arr1,arr2); $.parseXML(d);`
- Можливості мови (класи, ітератори) `$('.a').each(...); $.each(...);`
- Динамічне завантаження
- Незалежність від браузера
- Розширення

# Simple frameworks/libraries

## Переваги та недоліки

- 😊 Легко вивчити
- 😊 Маленький розмір
- 😊 Не нав'язує архітектуру
- 😞 Багато зусиль для оновлення даних/UI
- 😞 Немає механізмів взаємодії компонентів
- 😞 Складно створювати незалежні компоненти
- 😞 Зі збільшенням проекту швидко зростає складність

## 2. Frontend MVC Frameworks

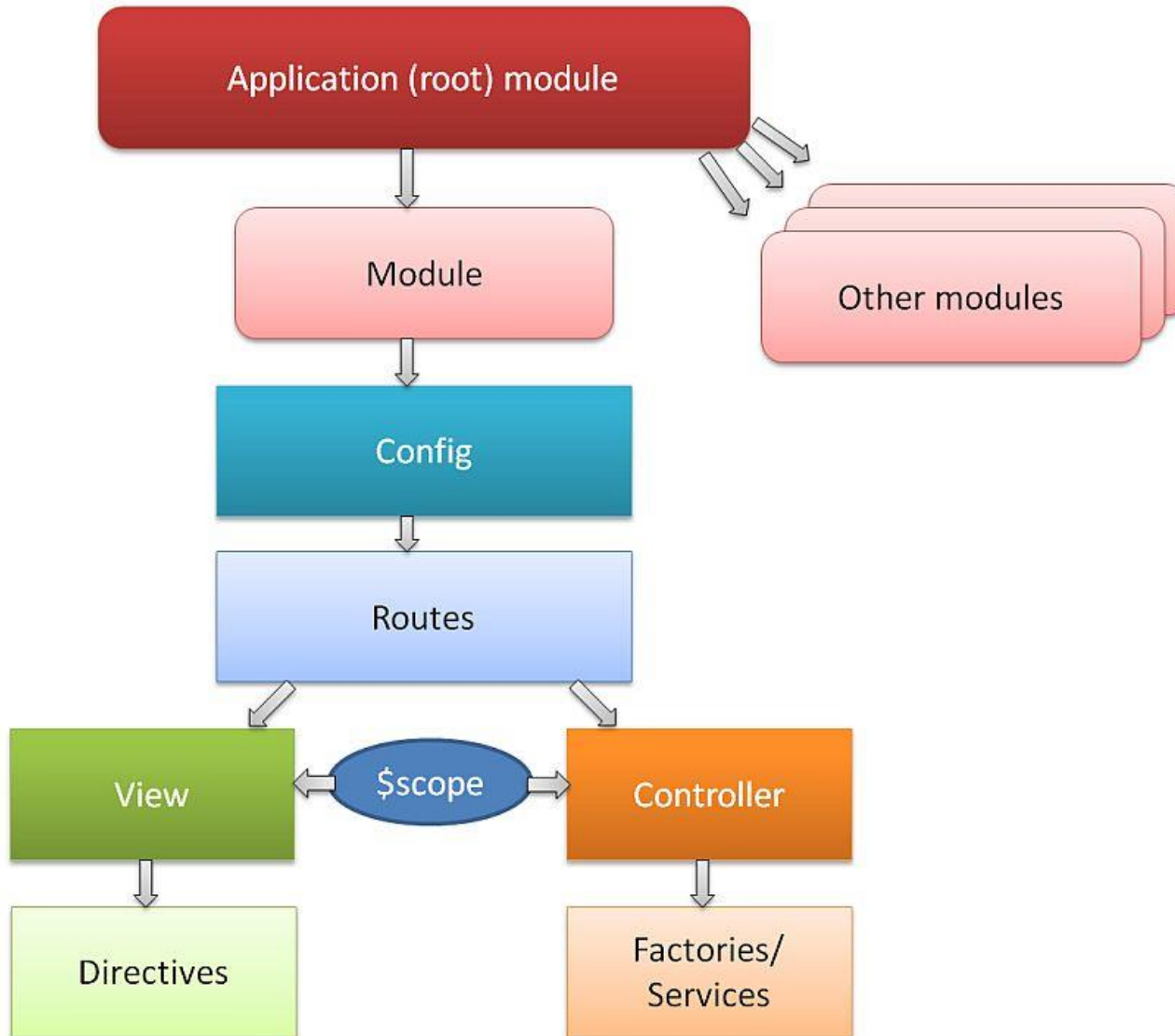


# Frontend MVC Frameworks

- Елементи UI прив'язані до елементів моделі
  - Two-way binding
- Template Engine
- Бізнес логіка в контролерах
- Стандартна архітектура
- Популярні приклади
  - [Angular](#), [Backbone](#), [Ember](#), [Knockout](#), ...
- <http://todomvc.com/>

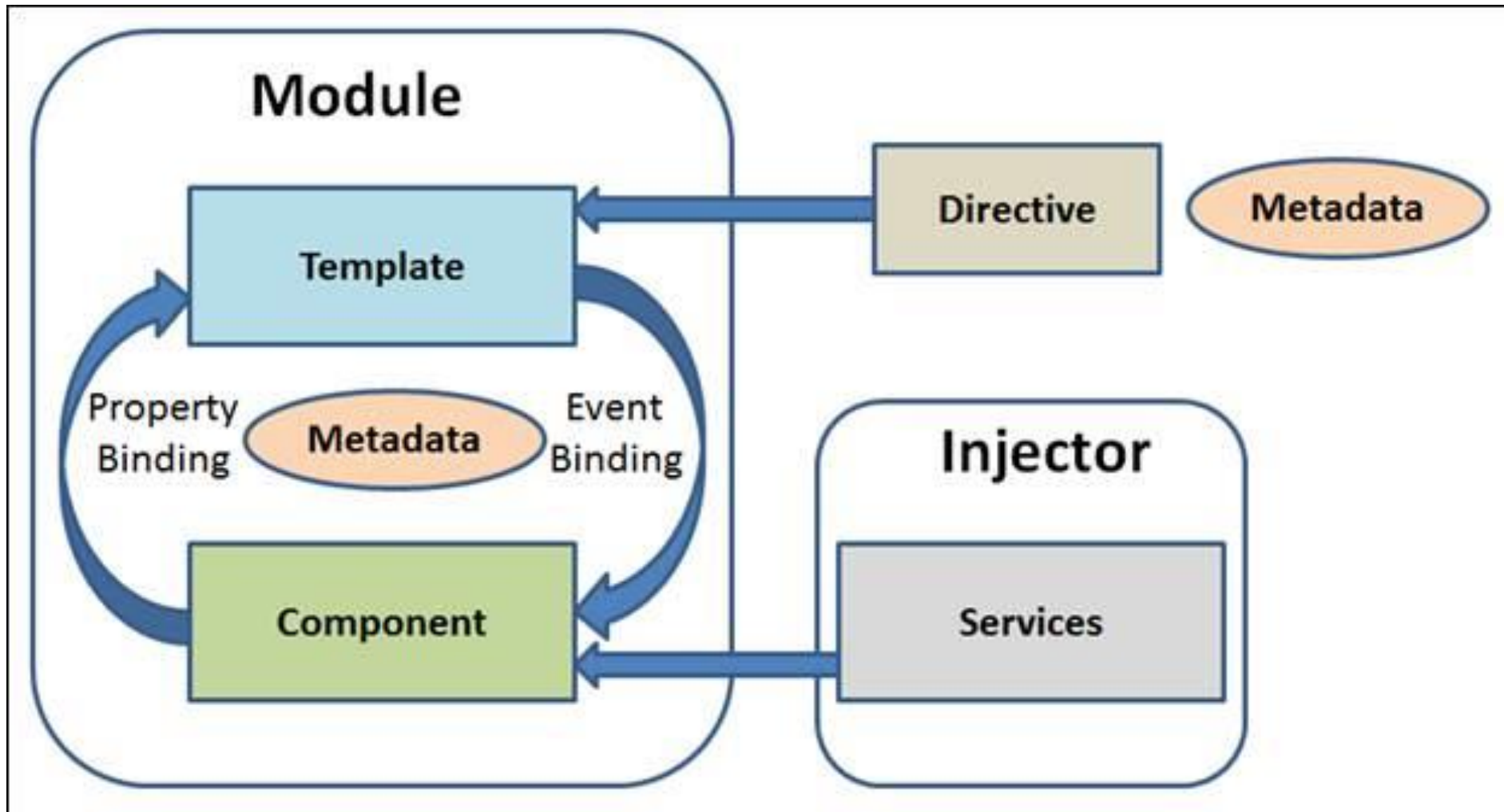


# Angular 1.x Architecture





# Angular 2 Architecture



# Frontend MVC Frameworks

## Переваги та недоліки

- 😊 Широка функціональність
- 😊 Two-way binding
- 😊 Структура коду
- 😊 Популярність, багато компонентів
- 😞 Складність вивчення
- 😞 Розмір, продуктивність
- 😞 Складно відлагоджувати
- 😞 Нав'язує архітектуру
- 😞 Погано працює в нестандартних випадках

# Можливості функціональних мов

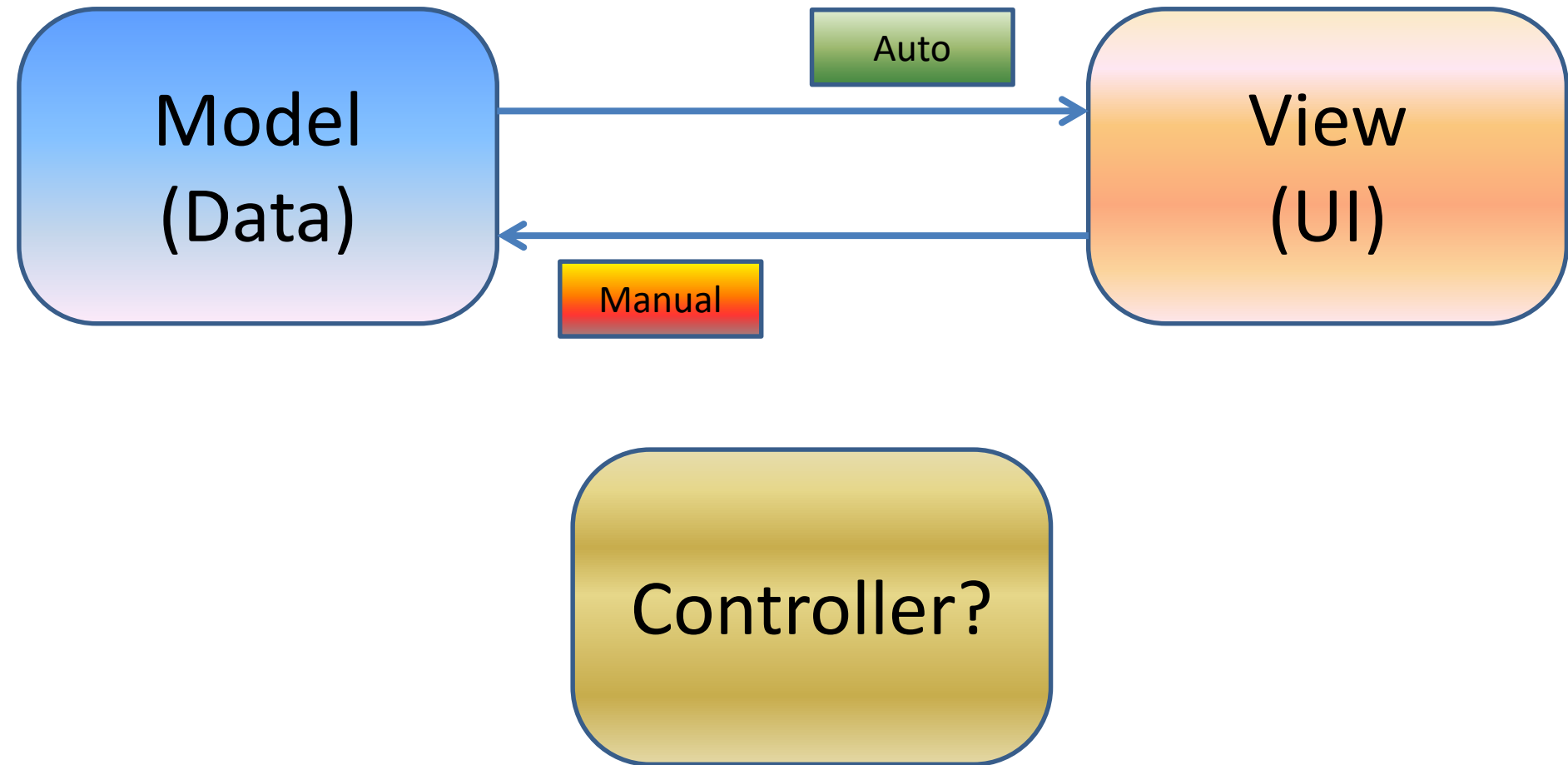
(на відміну від ОО)

- Функції як повноцінні елементи програми (first-class functions)
- Анонімні функції (lambda)
- Функції вищих порядків (high-order functions)
- Відсутність побічних ефектів (pure functions = no side effects)
- Незмінювані структури дані (immutable data structures)
- Більш потужна система типів
- Вивід типів (type inference)
- Lazy evaluation

# UI vs. functional?

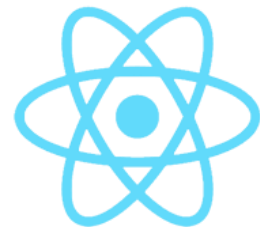
- Інтерфейс користувача добре описується об'єктно-орієнтованою парадигмою
  - Elements -> Objects
  - Events -> Messages
  - State encapsulation
- Функціональна парадигма краще описує роботу з даними
  - Data flow
  - Transformations
- Але іноді для UI також потрібна розумна робота з даними

### 3. One-way data flow



# One-way data flow

- View engine only – “V in MVC”
- Елементи UI генеруються з елементів моделі
  - One-way data binding
- Обробники подій
  - Змінюють модель
  - Зміни в моделі викликають зміну інтерфейсу
- Ефективні зміни інтерфейсу
  - Virtual DOM
- Приклади
  - [React](#), [Riot](#), [Vue](#), [Mercury](#)



React

# JSX – JS or XML?

- Створення JavaScript об'єктів з XML-подібним СИНТАКСИСОМ
- JSX

```
var url = "https://facebook.github.io/react/";  
var linkElement = <a className="btn" href={url}>Hello!</a>  
var myElement = <MyComponent someProperty={true} />;
```

- JavaScript

```
var url = "https://facebook.github.io/react/";  
var linkElement = React.createElement("a",  
  { className: "btn", href: url }, "Hello!");  
var myElement = React.createElement(MyComponent,  
  { someProperty: true });
```

- HTML

```
<a class="btn" href="https://facebook.github.io/react/"  
  data-reactid=".0.0"> Hello!</a>  
<div data-reactid= ".0.1"> ... </div>
```

- <https://babeljs.io/repl/>

# JSX – JS or XML?

HTML element  
(lowercase)

Custom element  
(uppercase)

- **JSX**

```
var url = "https://facebook.github.io/react/";  
var linkElement =  
  <a className="btn" href={url}>Hello!</a>  
var myElement = <MyComponent someProperty={true} />;
```

- **JavaScript**

```
var url = "https://facebook.github.io/react/";  
var linkElement = React.createElement("a",  
  { className: "btn", href: url }, "Hello!");  
var myElement = React.createElement(MyComponent,  
  { someProperty: true })
```

Any JS expression  
as property value



# JSX – JS or XML?

- JSX

```
var url = "https://facebook.github.io/react/";  
var linkElement = <a className="btn"  
  href={url}>Hello!</a>  
var myElement = <MyComponent someProperty={true} />;
```

JS property name,  
not HTML attribute

Custom element  
rendered with  
HTML elements

- HTML

```
<a class="btn" href="https://facebook.github.io/react/"  
  data-reactid=".0.0"> Hello!</a>  
<div data-reactid= ".0.1"> ... </div>
```

Internal bookkeeping  
(removed in React 15)

Called on each  
data change

# Model->View: render()

Create custom  
element

```
var HelloWorld = React.createClass({  
  render: function() {  
    return (  
      <p>  
        Hello, <input type="text" placeholder="Your name here" />!  
        It is {this.props.date.toTimeString()}  
      </p>  
    );  
  }  
});
```

JSX template

Pass data  
changes

```
setInterval(function() {  
  ReactDOM.render(  
    <HelloWorld date={new Date()} />,  
    document.getElementById('example')  
  );  
, 500);
```

Render inside  
this element

# View->Model: state, events

- State vs. props
  - State: mutable, usually stored at top component
  - Props: immutable, propagated to each component
- Explicit state changes: `setState(newState)`
- Event handlers
- `shouldComponentUpdate()` optimization

# View->Model: state, events

```
var LikeButton = React.createClass({
  getInitialState: function() {
    return {liked: false};
  },
  handleClick: function(event) {
    this.setState({liked: !this.state.liked});
  },
  render: function() {
    var text = this.state.liked ? 'like' : 'haven\'t liked';
    return (
      <p onClick={this.handleClick}>
        You {text} this. Click to toggle.
      </p>
    );
  }
});

ReactDOM.render(
  <LikeButton />,
  document.getElementById('example')
);
```

Initial state

Change state

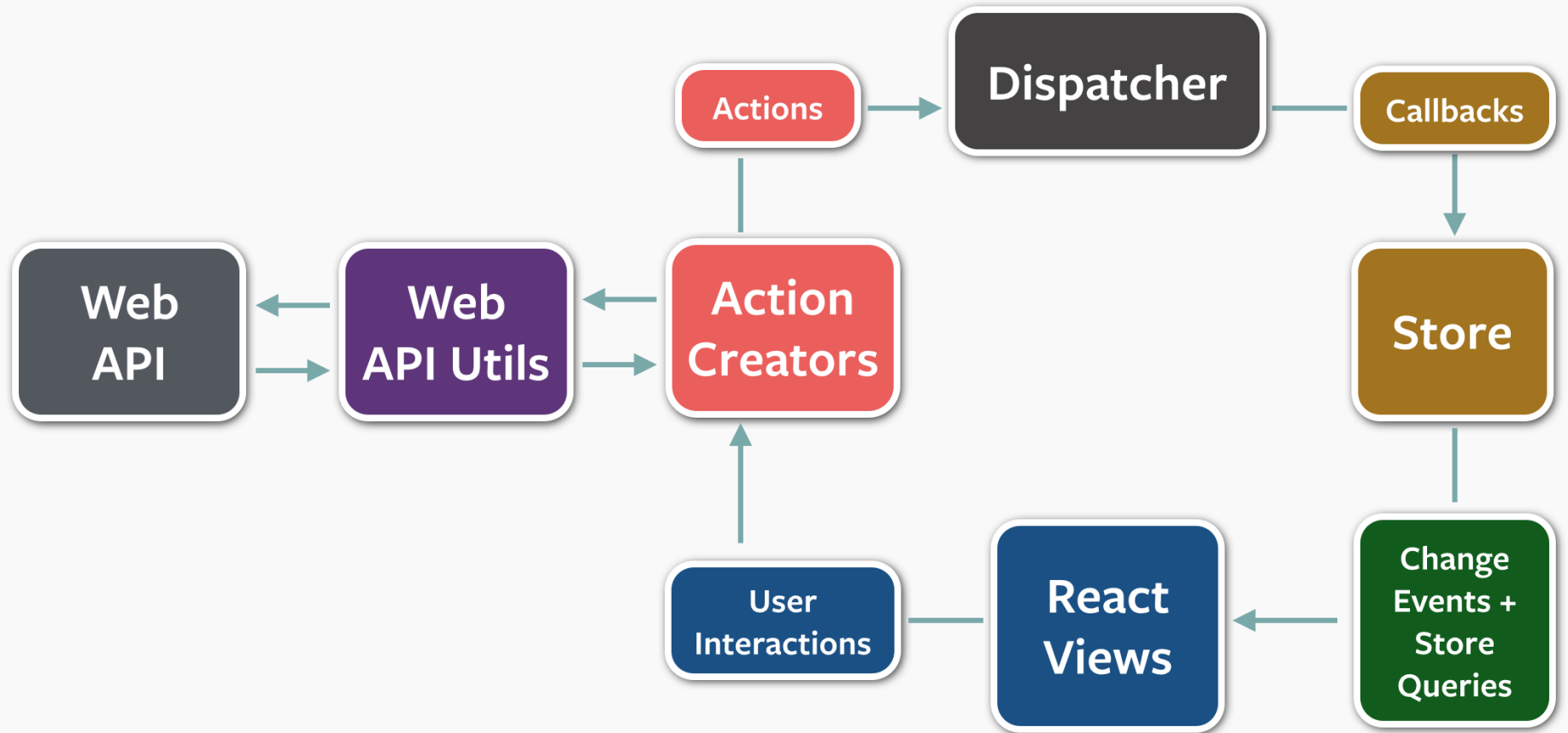
Event handler  
(JS property,  
not HTML attribute)

# One-way data flow

## Переваги та недоліки

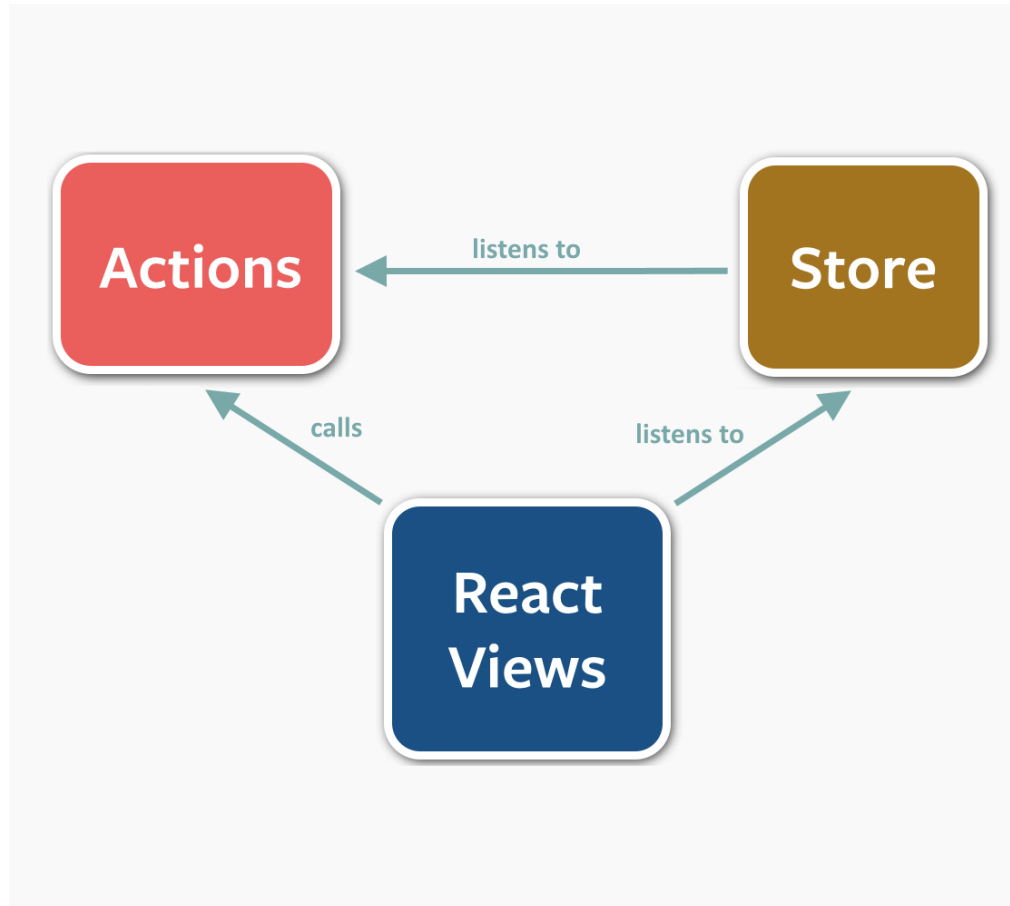
- 😊 Більш гнучкий підхід
- 😊 Сприяє розподілу стану та явній взаємодії компонентів
- 😊 Простіше відлагоджувати
- 😊 Продуктивність
- 😊 Набирає популярності
- 😞 Менше вбудованої функціональності
- 😞 Більше коду для оновлення моделі
- 😞 JavaScript замість HTML шаблонів
- 😞 Підтримка редакторів/IDE
- 😞 Проблеми з продуктивністю для великих списків

# Flux – архітектура для React



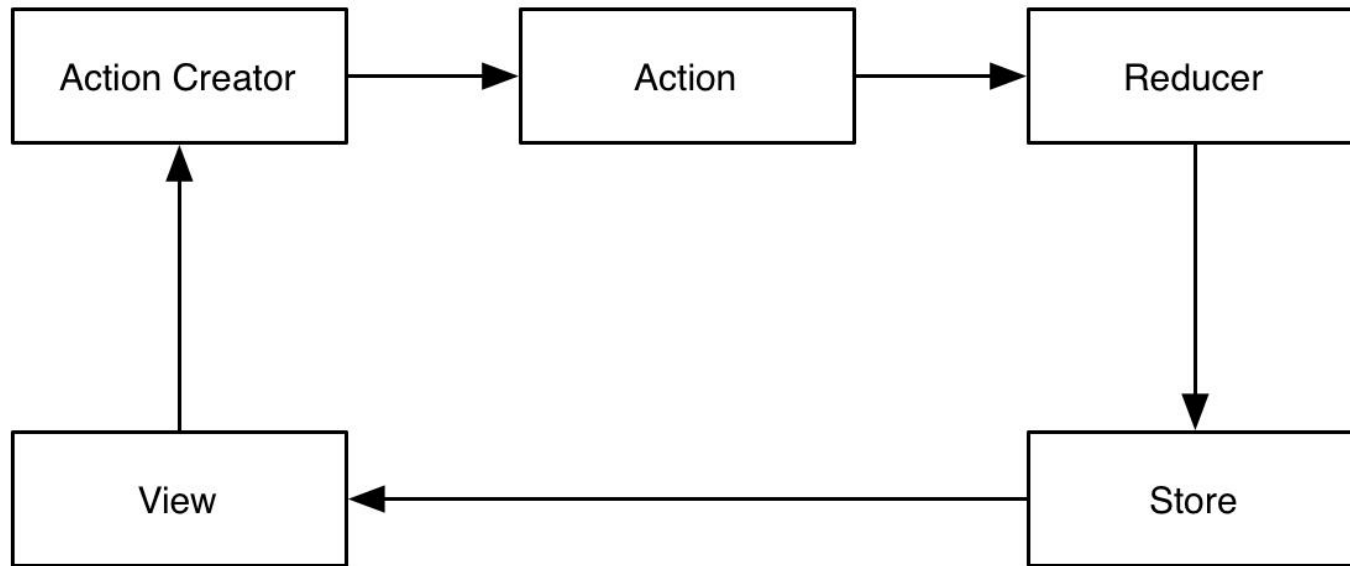
- Архітектура, а не фреймворк/бібліотека  
– Декілька реалізацій

# Reflux – реалізація Flux



- No Dispatcher
- Less boilerplate code
- Many helper mixins

# Redux – більше функціонального стилю



- Single store
- Reducer:  $(state, action) \Rightarrow state$ 
  - Pure function
  - Immutable state
- Reducer composition



# Elm – функціональна мова в браузері

- Inspired by Haskell
  - But less complexity
- Generates HTML, CSS, JavaScript
- Compile-time error detection
  - Static typing
  - Enforces pure functions
  - Modules with semantic versioning (detect breaking API changes)
- Interop with JavaScript
- Performance
- Аналоги: [ClojureScript](#) / [Om](#) , [PureScript](#)



# Polymer – незалежні компоненти

- Дозволяє описувати елементи DOM
  - Кожен елемент має внутрішню структуру
  - Стили всередині елемента незалежні від зовнішніх
- Реалізує нові стандарти
  - [Web Components](#)
  - [Custom Elements](#), [HTML templates](#), [HTML imports](#), [Shadow DOM](#)
- Бібліотека готових компонентів
  - Material Design
- Аналоги: [X-Tag](#), [Bosonic](#), [Angular 2](#), [Angular Material](#)

# Polymer custom element

```
<dom-module id="element-name">
  <template>
    <style>    /* CSS rules for your element */  </style>
    <!-- local DOM for your element -->
    <div>{{greeting}}</div> <!-- data bindings in local DOM -->
  </template>

  <script>
    // element registration
    Polymer({
      is: "element-name",
      properties: {
        // declare properties for the element's public API
        greeting: { type: String, value: "Hello! " }
      }
    });
  </script>
</dom-module>
```

# Universal=isomorphic apps

- Один код може виконуватись на клієнті і на сервері
  - Сервер: Node.js
  - Клієнт: React, Angular 2 Universal
- Підтримка клієнтів без JavaScript
- Підтримка пошукових машин

# Підтримка різних браузерів

- Відмінності
  - Обробка HTML
  - Властивості CSS
  - Доступні функції і можливості JavaScript



# Як забезпечити підтримку-1 (HTML/CSS)

- Раннє тестування в браузерях, які заплановані для підтримки
  - Прототипи найбільш складних або нових можливостей
  - Різні версії браузера



# Як забезпечити підтримку-2 (HTML/CSS)

- Писати код, що відповідає стандартам
- Валідатори
- Правильний DOCTYPE `<!DOCTYPE html>`
  - Підтримка валідаторів
  - Браузер працює в режимі відповідності стандартам, а не quirk mode
- Чистка стилів за замовчуванням
- Списки непідтримуваних конструкцій  
<http://caniuse.com/> <https://developer.mozilla.org/> ...
- Vendor prefixes -webkit-, -moz-, -o-, -ms-



# Як забезпечити підтримку-3 (HTML/CSS)

- Різні версії коду для різних браузерів
- Серверне вирішення
  - Генеруємо різні версії сторінки відповідно до браузера користувача
  - Як визначати браузер?
- JavaScript
  - Вставка, модифікація потрібних елементів
  - Класи, залежні від браузера
- Cross-browser frameworks (Bootstrap, ...)





# Як забезпечити підтримку-4 (HTML/CSS)

- Умовні коментарі (IE)
  - Розуміє тільки IE
  - Інші браузери ігнорують
- CSS hacks
  - Використання різних особливостей і помилок реалізації CSS в різних браузерах
  - Ненадійно, не відповідає стандартам

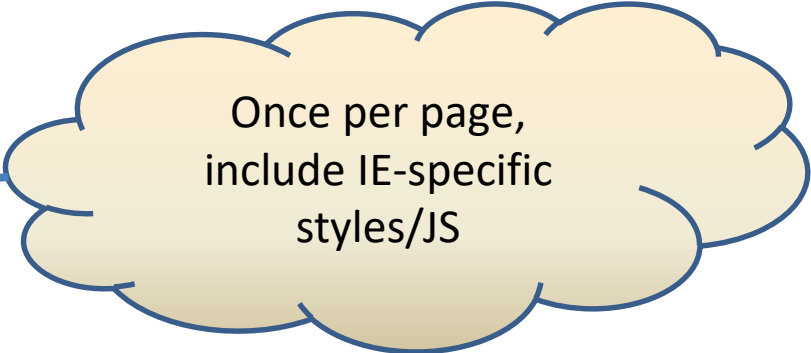
# Умовні коментарі

`<!--[if IE]>`

Fix IE bugs here

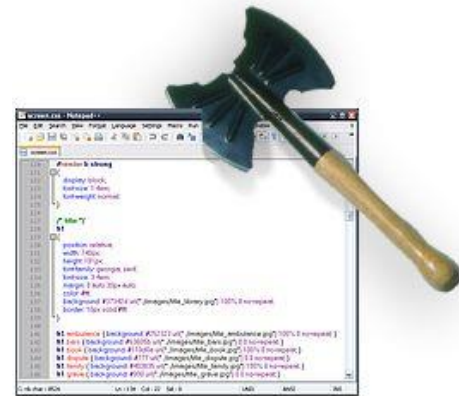
`<![endif]-->`

- `<!--[if IE 6]>`
- `<!--[if gt IE 6]>`
- `<!--[if lte IE 8]>`
- `<!--[if !IE]>`



Once per page,  
include IE-specific  
styles/JS

# CSS hacks



```
.box {  
    background: #00f; /* all browsers, Mac IE */  
    *background: #f00; /* IE 7 and below */  
    _background: #0f0; /* IE 6 and below */  
    _bac\kground: #f60; /* IE 6 only */  
}
```

Використовувати **небажано!!!**

# Підтримка різних браузерів (JavaScript)

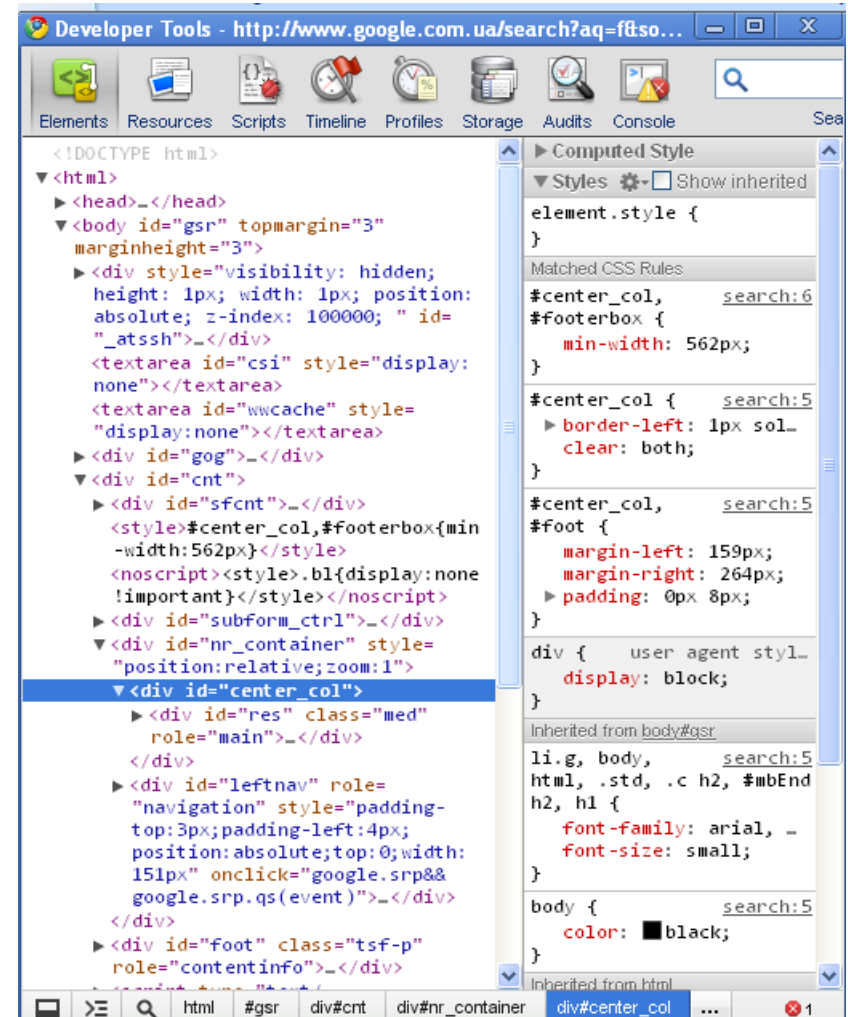
- Варіант 1: перевірка браузера і версії
  - `navigator.userAgent`
  - `navigator.vendor`
- Варіант 2: перевірка наявності необхідних об'єктів і властивостей (feature detection)
  - `if (window.XMLHttpRequest) { ... }`
  - Більш надійний спосіб
- Варіант 3: використання кросбраузерних бібліотек

# Використання JavaScript

- Динамічна модифікація сторінки (DOM)
- Валідація форм
- Асинхронна взаємодія з сервером (AJAX)
- Елементи керування в клієнтській частині
- Бізнес логіка
- Інші
  - Обчислення на клієнті
  - Графіка, анімація (canvas)

# Document Object Model (DOM)

- Представлення структури документа у вигляді набору об'єктів
- Дерево
- Пошук елемента
- Зміна властивостей або піделементів



# Доступ до елементів

- Через структуру дерева
  - `document.body`, `childNodes`, `parentNode`, ...
- Через `id`
  - `getElementById()`
- Пошук за тегом, ім'ям, класом
  - `getElementsByTagName()`, `getElementsByName()`, `getElementsByClassName()`
- Використання складних виразів (`selectors`, `XPath`)

# Selectors API

```
var el = document.querySelector('#test');  
var matches1 = el.querySelectorAll('div.highlighted > p');  
var matches2 = document.querySelectorAll('iframe[data-src]');
```


- Дозволяє використовувати селектори CSS
- Синтаксис звичний для розробників



# Використання XPath

```
var elems= document.evaluate("//div[@class='lh'] /  
    //div[@class='j'] /  
    //div[starts-with(@class,'story')]/h2[@class='title']",  
    document, null,  
    XPathResult.UNORDERED_NODE_SNAPSHOT_TYPE,  
    null);  
for (var i = 0; i < elems.snapshotLength; i++)  
    {  
        var elem=elems.snapshotItem(i);  
    }
```

# Модифікація елементів

- Атрибути
  - Властивості (id, className, style, href, ...)
  - `getAttribute()/setAttribute()`
- Елементи
  - `document.createElement()`
  - `appendChild()`
- innerHTML 
  - Можливість XSS
  - Для зміни тексту використовувати `textContent`

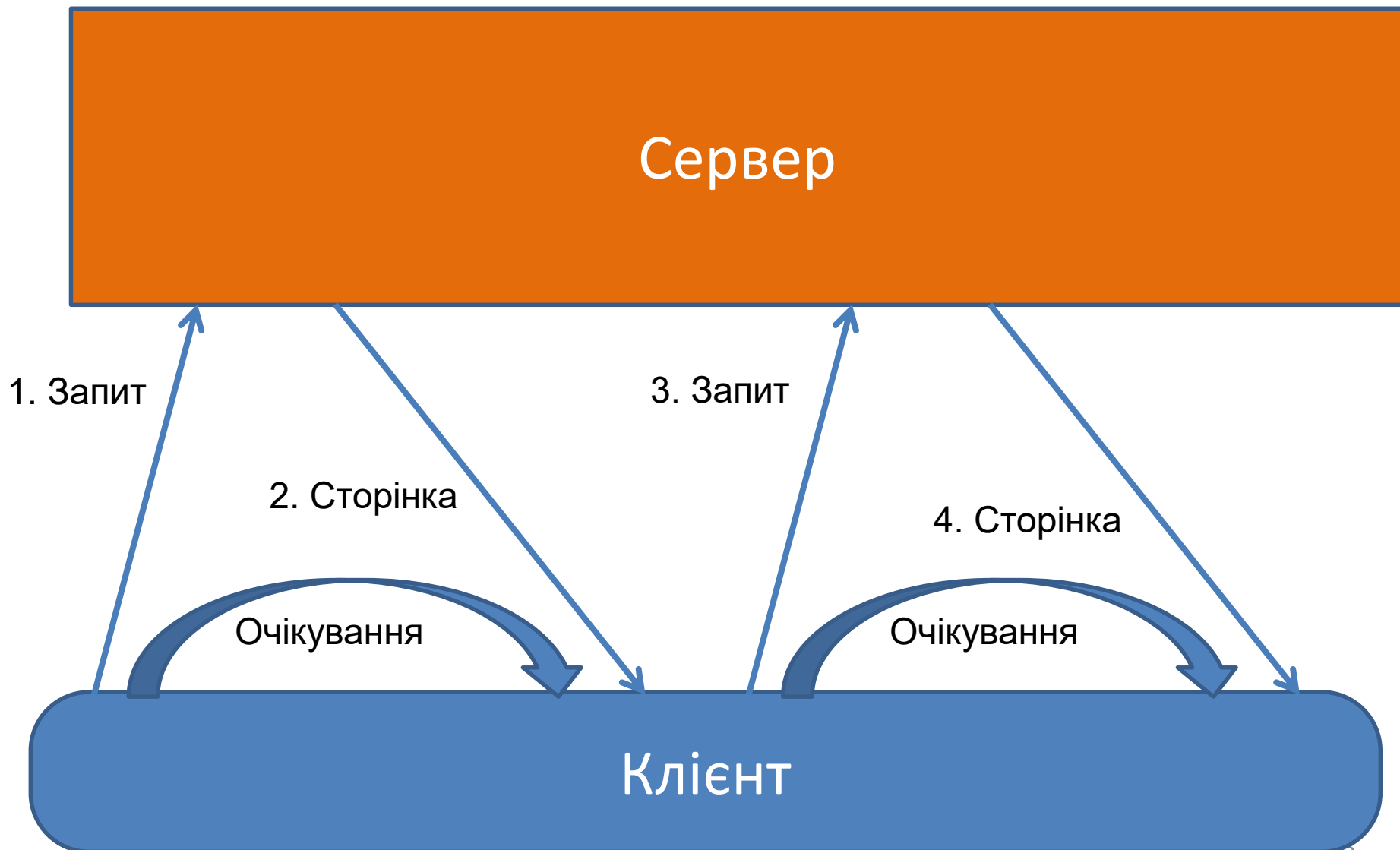
# Валідація форм

- Обробники подій (onkeyup, onfocus/onblur, onsubmit)
- Перевірка властивостей елемента форми (value)
- Повідомлення про помилку
- Не обмежуватись клієнтською валідацією

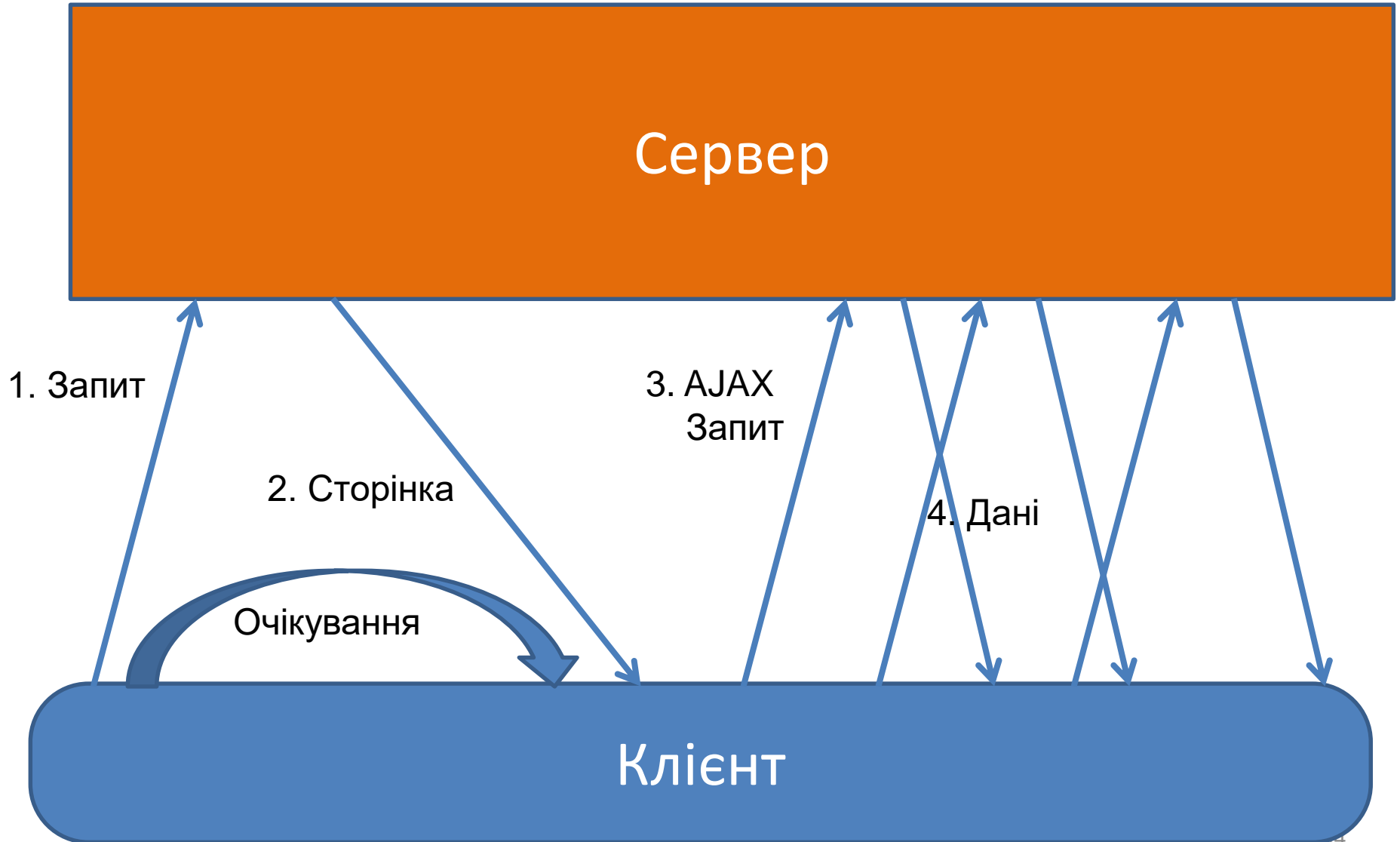
# Технологія AJAX

- Asynchronous JavaScript and XML
- Взаємодія з сервером без перезавантаження сторінки
- Асинхронна взаємодія
- Сервер передає дані, а не представлення

# Класичний веб-застосунок



# АЈАХ застосунок



# Об'єкт XMLHttpRequest

- Створення:
  - Стандарти: **new XMLHttpRequest()**
  - IE6: **new ActiveXObject("Microsoft.XMLHTTP")**
- Метод **open(httpMethod, url, isAsync)**
- HTTP заголовки: **setRequestHeader(name,value)**
- Передача запиту: метод **send(body)**
- Обробка результату: **onreadystatechange**
  - В XML: **req.responseXML**
  - В інших форматах: **req.responseText**



# Безпека

- Same origin policy: можна взаємодіяти тільки з даними, що надходять з того ж домену
  - <http://www.example.com:8080/dir/page.html>
- Не розповсюджується на елементи script, img, link, object
- Cross-Origin Resource Sharing (XHR level 2)
  - Підтримка на сервері за рахунок Access-Control-Allow-Origin
- HTML5: **postMessage**
  - Дозволяє передати повідомлення іншому веб-застосуванню
  - Отримувач вирішує, як обробляти повідомлення



# Формат передачі даних

- XML

- Більше підтримки в серверних технологіях
- Більш складний, накладні розходи

```
<Books>
  <Book ISBN="0553212419">
    <title>Sherlock Holmes: Complete Novels...
    <author>Sir Arthur Conan Doyle</author>
  </Book>
```

- JSON

- Проста мова
- Можливість обходити same origin policy
- Проблеми з безпекою

```
{
  "dict": {
    "key": ["activities", "katamarize gallery"],
    "array": {
      "works": {
        "client": "KATAMARI INC.",
```

- Інші варіанти

- HTML
- Plain text

# Переваги і недоліки AJAX

- 😊 Більша інтерактивність
- 😊 Менше даних передається з серверу
- 😞 Складність правильної реалізації асинхронної взаємодії
- 😞 Проблеми з історією перегляду
- 😞 Проблеми з закладками
- 😞 Підтримка пошукових систем, браузерів без JavaScript
- 😞 Збільшення кількості запитів до сервера
- 😞 Доступ до даних з іншого домену

# Висновки

- Багато різних конфігурацій браузерів – треба вміти всіх підтримувати
  - Бажано універсальним чином, а не хаками для кожної версії
- Чіткий розподіл структури документу (HTML), зовнішнього вигляду (CSS) та поведінки (JavaScript)
- Використовувати сучасні інструменти та бібліотеки
- Можна підглядіти, як реалізовано в інших

# Implementation Checklist – MUST

- ☐ M3.1. Separate document structure (HTML) and appearance (CSS)
- ☐ M3.2. Use `<!DOCTYPE html>`
- ☐ M3.3. Add `<meta name="viewport">` with appropriate parameters
- ☐ M3.4. Never execute external or user-supplied JavaScript code
  - ☐ M3.4.1. `eval()`
  - ☐ M3.4.2. Dynamic `<script>` elements
  - ☐ M3.4.3. `innerHTML`

# Implementation Checklist – SHOULD

- ☐ S3.1. Don't use table layout (unless representing table)
- ☐ S3.2. Use meaningful names for classes/IDs in CSS
- ☐ S3.3. Minify and combine CSS files to reduce page load time
- ☐ S3.4. Use caching on CSS files (e.g. 1 year)
- ☐ S3.5. Check your HTML and CSS code for older snippets of code copied from previous projects / other sites
  - ☐ S3.5.1. Unneeded vendor prefixes
- ☐ S3.6. Clear default styles on all relevant elements
- ☐ S3.7. Consider using cross-browser frameworks or polyfill instead of manually adding support for older browsers
- ☐ S3.8. Use validators to check for common errors
- ☐ S3.9. Use IE conditional comments only to include additional libraries for old IE versions; don't use it throughout code
- ☐ S3.10. Avoid CSS hacks

# Implementation Checklist – SHOULD

- ☐ S3.11. Consider using CSS preprocessors
- ☐ S3.12. Use responsive grid layout (e.g. framework or generator)
- ☐ S3.13. Don't restrict zooming
- ☐ S3.14. Check how your site behaves under unusual conditions
  - ☐ S3.14.1. Screen size
  - ☐ S3.14.2. Pixel density
  - ☐ S3.14.3. Zoom
- ☐ S3.15. Avoid using plugins (Flash, Silverlight, Java Applets, ...)

# Implementation Checklist – SHOULD

- ☐ S3.16. Use JavaScript frameworks/libraries to simplify development
- ☐ S3.17. Use JavaScript tools
  - ☐ S3.17. 1. Static analysis
  - ☐ S3.17. 2. Unit testing
  - ☐ S3.17. 3. Documentation
- ☐ S3.18. Enable strict mode (“use strict”;
- ☐ S3.19. Use `textContent` instead of `innerHTML` to set element text
- ☐ S3.20. Avoid complex DOM changes
  - ☐ S3.20.1. Use `DocumentFragment` to make changes outside of DOM
- ☐ S3.21. Validate user inputs on client when possible
- ☐ S3.22. Use CORS to make cross-origin AJAX calls
- ☐ S3.23. Minify and combine JavaScript files
- ☐ S3.24. Use caching on JavaScript files (e.g. 1 year)
- ☐ S3.25. Use feature detection instead of parsing `userAgent`

# References

- Feature support in browsers <http://caniuse.com/>
- Browser compatibility  
<http://www.quirksmode.org/>
- Mozilla Developer Network  
<https://developer.mozilla.org/>
- <https://css-tricks.com/>
- HTML/CSS/JS playgrounds  
<http://www.sitepoint.com/7-code-playgrounds/>



# References

- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction to Object-Oriented JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Introduction_to_Object-Oriented_JavaScript)
- [http://www.w3.org/wiki/JavaScript best practices](http://www.w3.org/wiki/JavaScript_best_practices)
- <http://eloquentjavascript.net/>
- <http://htmlbook.ru/>
- <http://learn.javascript.ru/>

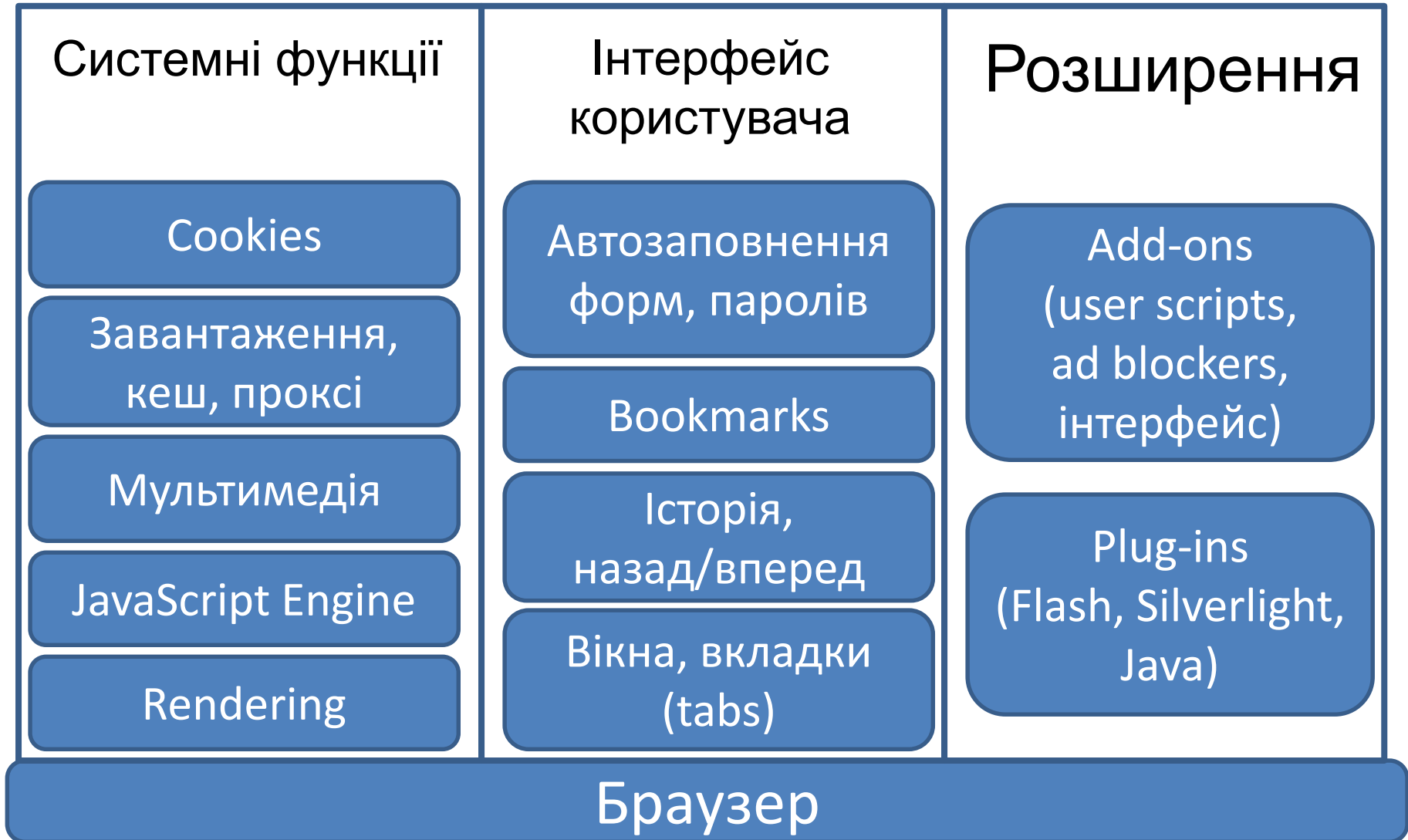
# References

- <https://github.com/sorrycc/awesome-javascript>
  - <https://github.com/sotayamashita/awesome-css>
  - <https://github.com/emijrp/awesome-awesome>
- <https://www.reddit.com/r/Frontend/>
  - <https://www.reddit.com/r/javascript/>
  - <https://www.reddit.com/r/css/>
  - <https://www.reddit.com/r/html5/>



Old slides

# Компоненти клієнта

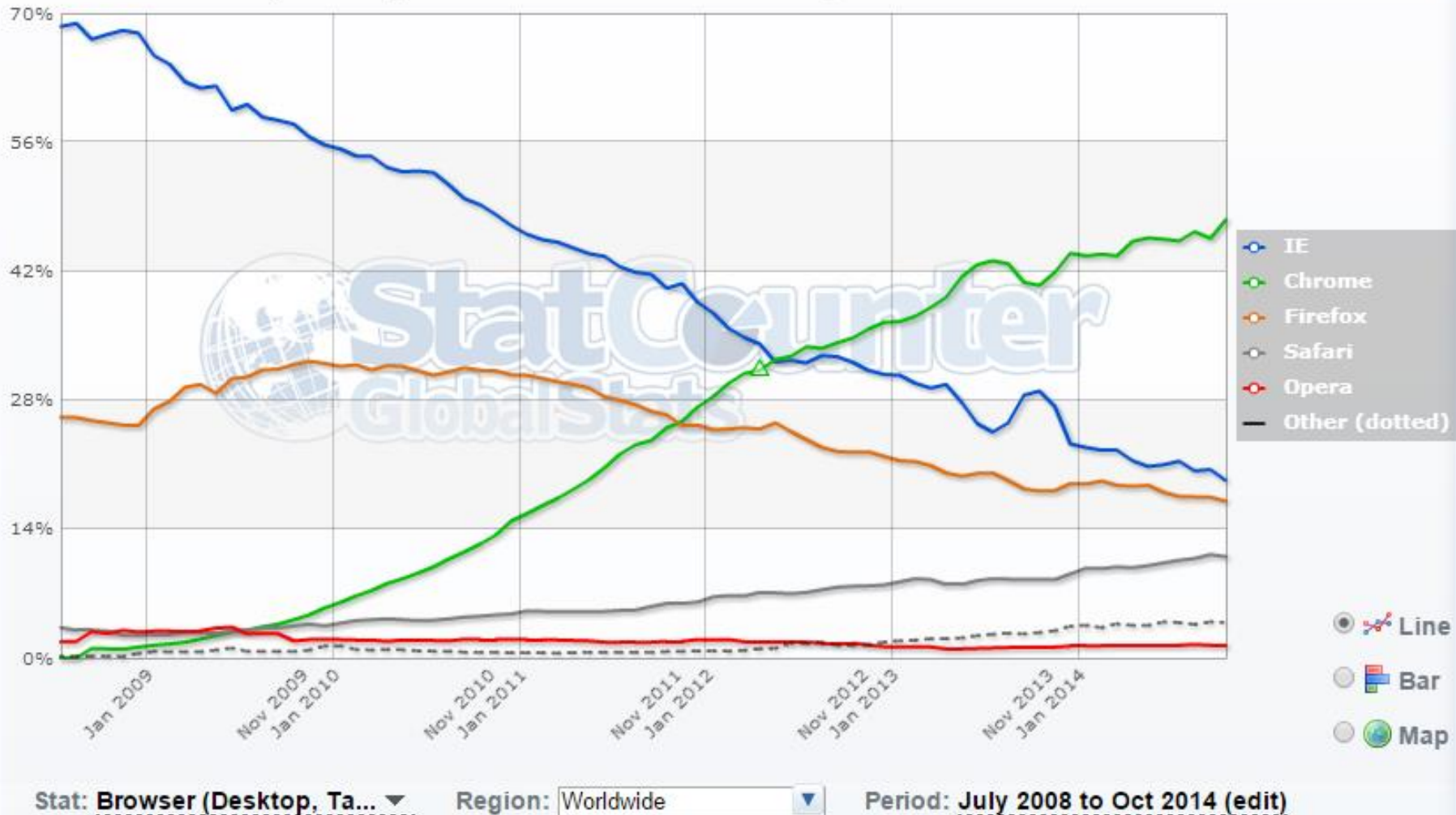


# Популярні браузери

Браузер	Rendering Engine	JavaScript engine
Internet Explorer Edge	Trident EdgeHTML	JScript Chakra (9)
Firefox	Gecko	SpiderMonkey TraceMonkey(3.5) JägerMonkey (4) IonMonkey(18+) +OdinMonkey(22+)
Chrome	WebKit Blink (28+)	V8
Safari	WebKit	JavaScriptCore SquirrelFish (4) Nitro
Opera	Presto(1-12)  WebKit(14)->Blink(15+)	Futhark Carakan(10.50) V8(14+)

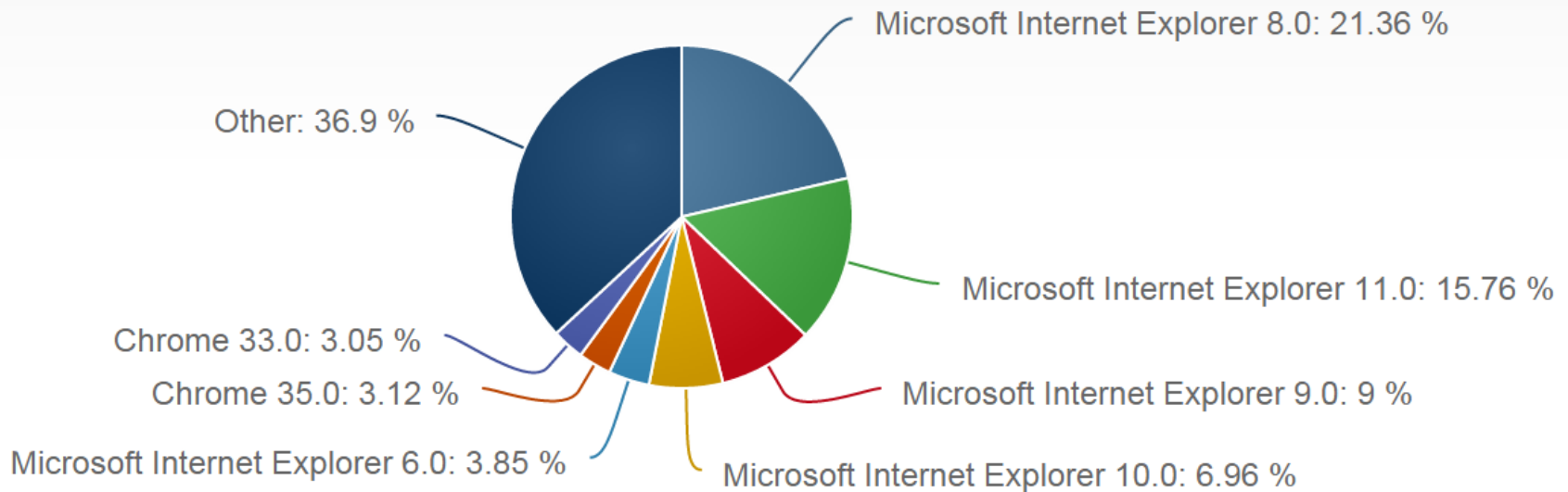
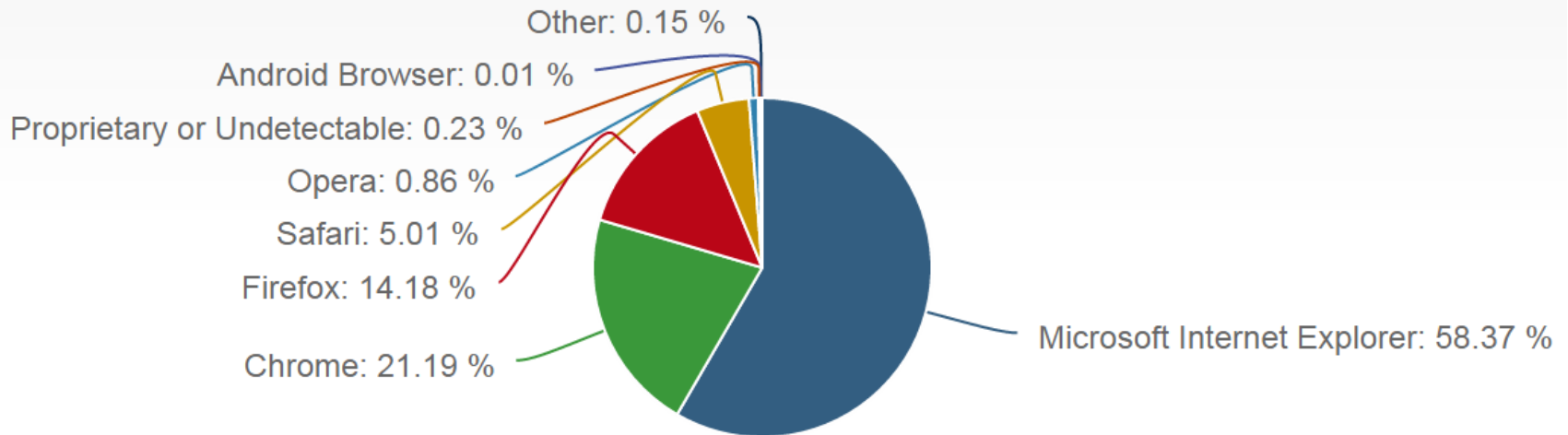
# Market Share

StatCounter Global Stats  
Top 5 Desktop, Tablet & Console Browsers from July 2008 to Oct 2014



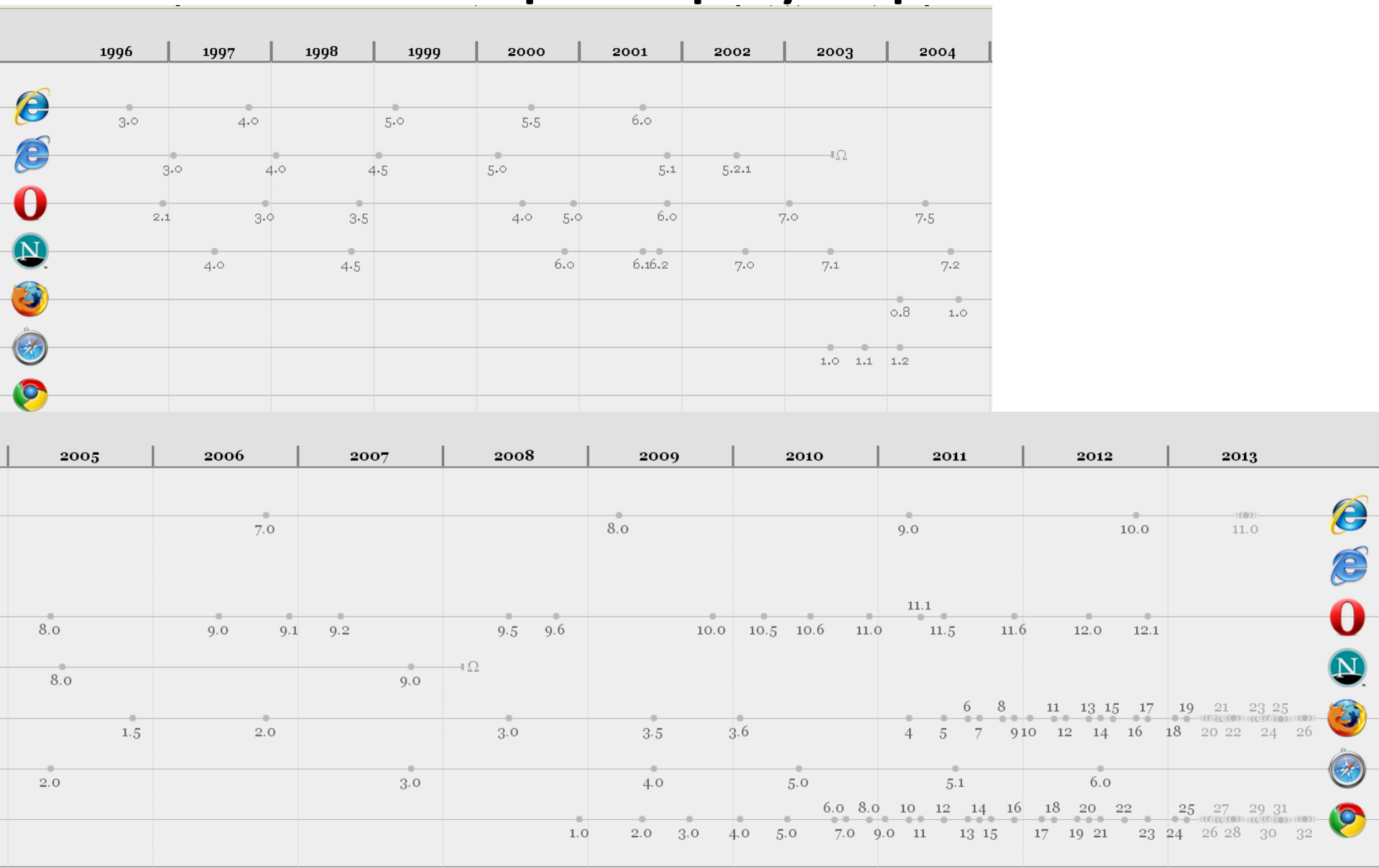
# Market Share

## Net Applications, September 2014





# Історія браузерів



# DOCTYPE

Version	DOCTYPE
HTML 4.01 Strict	<  TYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 Transitional	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/transition.dtd"> 
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 Strict	<  tn<img alt="Intel logo" data-bbox="375 535 475 625"/>HTML PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">   
XHTML 1.0 Transitional	<  PUBLIC "-//W3C//DTD  ansitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"> 
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
HTML 5	<  >    

# DOCTYPE

Version	DOCTYPE
HTML 4.01 Strict	<!DOCTYPE <b>CAYENNE</b>  3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
HTML 4.01 Transitional	<!DOCTYPE  BLI  ED HTML 4.01   "http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 Strict	<  tn   XHTML  EN  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
XHTML 1.0 Transitional	<!DOCTYPE  C "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">      
HTML 5	<!DOCTYPE    



# DOCTYPE

Version	DOCTYPE
HTML 4.01 Strict	 CTYPI <b>CAYENNE</b>  3C//DTD HTML 4.01//EN" strict.dtd">
HTML 4.01 Transitional	 BLI  TD  Transitional//EN" http://www.w3.org/.../loose.dtd
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"> 
XHTML 1.0 Strict	<?xml PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">    
XHTML 1.0 Transitional	<?xml PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">   
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 
HTML 5	<!DOCTYPE html>       

# DOCTYPE




Version	DOCTYPE
HTML 4.01 Strict	   3C//DTD HTML 4.01//EN" strict.dtd">
HTML 4.01 Transitional	  TD HTML 4.01 Transitional//EN" http://www.w3.org/TR/html4/loose.dtd">
HTML 4.01 Frameset	<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
XHTML 1.0 Strict	  XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"> 
XHTML 1.0 Transitional	  C "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
XHTML 1.0 Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
XHTML 1.1	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"> 
HTML 5	      

# DOCTYPE

Version	DOCTYPE
HTML 4.01 Strict	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN" "http://www.w3.org/TR/html4/strict.dtd"&gt;</pre> 
HTML 4.01 Transitional	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"&gt;</pre> 
HTML 4.01 Frameset	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"&gt;</pre>
XHTML 1.0 Strict	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"&gt;</pre> 
XHTML 1.0 Transitional	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"&gt;</pre> 
XHTML 1.0 Frameset	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"&gt;</pre>
XHTML 1.1	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"&gt;</pre>
HTML 4.0	



# DOCTYPE

Version	DOCTYPE
HTML 4.01 Strict	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd"&gt;</pre>  
HTML 4.01 Transitional	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional "http://www.w3.org/TR/html4/loose.dtd"&gt;</pre>  
HTML 4.01 Frameset	<pre>&lt;!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd"&gt;</pre>
XHTML 1.0 Strict	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd"&gt;</pre> 
XHTML 1.0 Transitional	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"&gt;</pre>   
XHTML 1.0 Frameset	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd"&gt;</pre>
XHTML 1.1	<pre>&lt;!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd"&gt;</pre>
HTML 4.0	<p>DON'T USE!</p>   

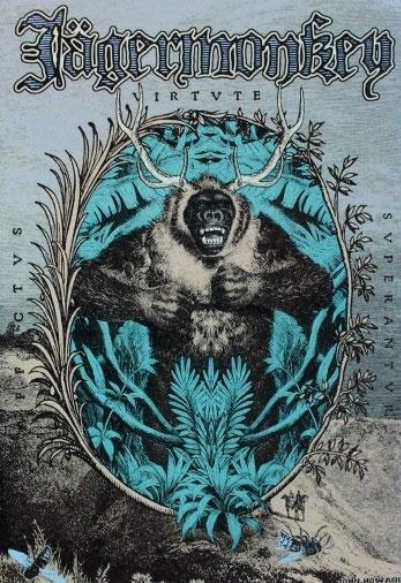
# Статичні та динамічні сторінки

- Статичні сторінки (тонкий клієнт)
  - HTML
  - CSS
- Динамічні сторінки (товстий клієнт)
  - HTML
  - CSS
  - JavaScript



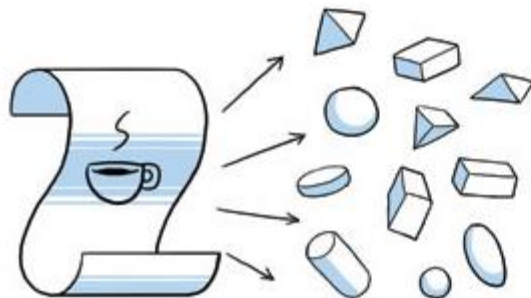
# JavaScript frameworks

© W3Techs.com	usage 24 Oct 2014	usage 1 Nov 2013	usage 1 Nov 2012	usage 1 Nov 2011
<b>None</b>	35.8%	38.6%	41.2%	51.4%
<b>jQuery</b>	60.7%	56.8%	52.6%	40.0%
<b>Modernizr</b>	6.8%	4.0%		
<b>MooTools</b>	4.4%	5.1%	5.0%	5.2%
<b>Prototype</b>	2.4%	2.8%	3.8%	4.5%
<b>ASP.NET Ajax</b>	2.2%	2.3%	3.0%	2.8%
<b>Script.aculo.us</b>	2.0%	2.2%	2.9%	3.3%
<b>YUI Library</b>	0.7%	0.9%	1.5%	3.7%



# JavaScript, DOM, AJAX

JAVASCRIPT ITSELF IS **CLASSLESS**.  
YOU CAN CREATE A NEW OBJECT,  
DYNAMICALLY ADD PROPERTIES TO  
IT AND GO ON.



BUT IN **V8**, AS EXECUTION GOES ON,  
OBJECTS THAT END UP WITH THE SAME  
PROPERTIES WILL SHARE THE SAME HIDDEN  
CLASS AND WE CAN START APPLYING  
DYNAMIC OPTIMIZATIONS BASED ON THAT.

