**CSS-фреймворк** — фреймворк, созданный для упрощения работы верстальщика, быстроты разработки и исключения максимально возможного числа ошибок вёрстки (проблемы совместимости различных версий браузеров и т. д.). Как и библиотеки скриптовых языков программирования, CSS-библиотеки, обычно имеющие вид внешнего css-файла, «подключаются» к проекту (добавляются в заголовок веб-страницы).

**Преимущества**

1. Позволяет не искушенному в тонкостях вёрстки программисту или дизайнеру правильно создать HTML-макет.
2. Вёрстка на базе слоёв, а не таблиц.
3. Более быстрая разработка.
4. Кроссбраузерность.
5. Возможность использования генераторов кода и визуальных редакторов.
6. Единообразие кода при работе в команде позволяет снизить число разногласий при разработке.

**Недостатки**

1. Библиотеки бывают сильно «раздуты» — может быть много лишнего кода, который не будет использоваться в проекте.
2. Дизайн будет зависеть от css-библиотеки.

**Отличия**

Некоторые библиотеки используют сброс стилей (reset.css) для приведения в соответствие стилей по умолчанию для разных браузеров, в то время как другие используют нормализацию (Normalize.css), что уменьшает количество изначального кода библиотеки.

**Типы CSS-библиотек**

Из-за упомянутых недостатков — использование CSS-библиотек вызывает споры в профессиональном сообществе. Также это привело к появлению различных типов CSS-библиотек. Условно можно выделить два типа: *Всеобъемлющие и Ограниченные*. Третьим вариантом может быть разработка собственной библиотеки. Этот вариант предпочитает большинство разработчиков, так как это дает выгоды персонального решения и уменьшает негативные моменты зависимости от использования сторонних библиотек.

**Всеобъемлющие CSS-библиотеки**

Этот тип библиотеки пытается охватить большинство вещей, которые могут понадобиться разработчику. К этому типу отнесем библиотеки, которые включают CSS для верстки и сброса (или какую-то основу).

**Ограниченные CSS-библиотеки**

Как следует из названия, библиотеки этого типа охватывают лишь ограниченный объём потребностей или имеют конкретную цель.

**Примеры CSS-библиотек**

***Всеобъемлющие***

1. Maxmertkit — модульная библиотека с множеством тем, виджетов и javascript-компонентов; предназначена для разработки адаптивных веб-интерфесов.
2. Twitter Flight — библиотека, созданная разработчиками Twitter.
3. Blueprint — наиболее популярная библиотека.
4. 960 Grid System — библиотека для быстрой разработки макетов.
5. Yet Another Multicolumn Layout (YAML) — имеет инструмент генерации кода, возможность создания адаптивных интерфейсов.
6. Bluetrip
7. Bootstrap — библиотека, созданная разработчиками Twitter.
8. css-framework — российская библиотека для разработки веб-интерфейсов.
9. Foundation — продвинутая библиотека для разработки адаптивных интерфейсов.
10. Fluid Baseline Grid System
11. Golden Grid System — библиотека для разработки адаптивных интерфейсов на основе золотого сечения.
12. Gumby Framework
13. The 1140px CSS Grid System
14. Topcoat - небольшая библиотека от Adobe распространяется по лицензии Apache 2.0

***Ограниченные***

1. jQuery UI CSS Framework
2. CSS-based Slideshow System — библиотека для создания презентаций.

**SEMANTIC UI**

**Types of Components**

The following table helps differentiate between different types of components in Semantic UI's documentation.

**Globals** are styles that are applied across a site. These include things like css resets, and sitewide font, link and sizing defaults. Most importantly, globals include site-wide theming variables that other components can inherit and modify.

**Element**. UI elements are page elements with a single function. They can exist alone or in a plural form with elements sharing qualities.

**Collections** are heterogenous groups of components which are usually found together. They describe a list of "usual suspects" which appear in a certain context. They may include and extend other ui elements for use in certain contexts—for example form may extend ui dropdown or input—as well as include their own content.

**Views.** A view is a convention for presenting specific types of content that is usually consistent across a website. These include things like comments, user activity feeds, or cards.

**Modules** are components that include both a definition of how they appear and how they behave. These include components like, accordions, dropdowns, and popups.

**Behaviors** are standalone javascript components that describe how page elements should act, but not how they should appear. Behaviors include things like form validation, state management, and API request routing.

<button class="ui button">

Button

</button>

<div class="ui button" tabindex="0">

Focusable

</div>

Foundation