**Каскадность и специфичность — “магия” стилей, которую должен понимать каждый**

**2.1. Что такое каскадность**

CSS расшифровывается как “Каскадные таблицы стилей” (Cascading Style Sheets).  
**Каскадность** — это принцип, по которому браузер решает, какой стиль применить, если один и тот же элемент попадает под разные правила.  
**Каскад** — это “столкновение потоков”, когда несколько стилей борются за контроль над одним элементом.

**Пример:**

css

КопироватьРедактировать

p { color: blue; }

.main-text { color: green; }

#unique { color: red; }

Если параграф имеет и класс, и id:

html

КопироватьРедактировать

<p id="unique" class="main-text">Текст</p>

— он будет красным, потому что id “сильнее”.

**2.2. Порядок применения стилей**

1. **Браузерные стили** (user agent stylesheet): дефолтное оформление (Times New Roman, синий подчеркиваемый ссылки и т.д.).
2. **Внешние таблицы стилей** (<link rel="stylesheet">)
3. **Встроенные стили** (<style>)
4. **Инлайн-стили** (style="color:...")
5. **!important** — “убийца каскада”, если очень нужно перебить всё.

**Важно:** Чем “ближе” стиль к элементу, тем выше его приоритет.

**2.3. Специфичность (specificity)**

**Специфичность** — “вес” селектора. Чем “уникальнее” правило, тем оно “сильнее”.

* **Inline-стиль**: 1000 баллов
* **ID**: 100 баллов
* **Класс/атрибут/псевдокласс**: 10 баллов
* **Тег/псевдоэлемент**: 1 балл

**В реальной жизни:**

css

КопироватьРедактировать

p { color: blue; } /\* 1 \*/

.main-text { color: green; } /\* 10 \*/

#unique { color: red; } /\* 100 \*/

<p id="unique" class="main-text"> будет красным.

**Лайфхак:**

* !important = 10000, но старайся не использовать (кроме “реанимации”).

**2.4. Конфликты и “бои” стилей**

**Классика:**

css

КопироватьРедактировать

.btn { color: blue; }

.btn { color: green; }

Второе правило сработает — потому что оно ниже по коду.

**А теперь усложним:**

css

КопироватьРедактировать

button { color: red; }

.btn { color: blue; }

#main .btn { color: orange; }

.btn.important { color: purple; }

— Чей цвет будет у <button class="btn important">?  
**Ответ:** purple (двойной класс сильнее одиночного).

**2.5. Практические кейсы**

* Почему иногда цвет “не работает”?  
  — Скорее всего, его перебил более специфичный селектор.
* Почему стили “слетают” после подключения сторонней библиотеки?  
  — Их селекторы оказались более “сильными”.
* Почему иногда !important “не спасает”?  
  — Возможно, inline-стиль с !important всё равно сильнее.

**2.6. Примеры из реальных проектов**

* Большие проекты делят CSS-файл на секции, чтобы избежать конфликтов (по методологии BEM, Atomic CSS, и др.).
* В больших командах договариваются:  
  — “Не используем id для стилей!”  
  — “Не применяем !important без крайней нужды.”
* Современные фреймворки (например, Bootstrap, Tailwind) генерируют очень специфичные классы, чтобы перебивать всё на сайте — поэтому “свои” стили приходится писать либо после их подключения, либо тоже повышать специфичность.

**2.7. Best/Anti-practices и советы**

**Best:**

* Всегда используй классы для стилизации (а не id).
* Разделяй структуру (HTML) и оформление (CSS).
* Пиши стили в порядке “от общего к частному”.
* Минимизируй количество !important.

**Anti:**

* Не делай так:

css

КопироватьРедактировать

body \* { color: red !important; }

— “убьёшь” весь каскад.

* Не смешивай классы и id без необходимости.
* Не доверяй библиотекам — всегда тестируй свой CSS после их подключения.

**2.8. Ошибки новичков**

* Ожидание, что “любой стиль сработает всегда”.
* Перепутал селектор класса и id (.box vs #box).
* Использовал !important везде — потом ничего не перебить.

**2.9. Глоссарий**

| **Термин** | **Определение** |
| --- | --- |
| Каскадность | “Столкновение” стилей в одном элементе |
| Специфичность | “Вес” селектора, определяет, кто “выиграет” |
| !important | Супер-приоритет, перебивает всё, кроме inline-style |
| Inline-style | Стиль, написанный прямо в теге |

**2.10. Современные фишки и будущие стандарты**

* Новые фичи:  
  :is(), :where() — позволяют “группировать” селекторы, упрощают код, но специфичность минимальна.
* Компонентный CSS (CSS-in-JS) — у каждого компонента свой “локальный” стиль, нет конфликтов.

**2.11. Вопросы для самопроверки**

1. Какой стиль “сильнее” — .card или #card?
2. Почему иногда !important “не спасает”?
3. Что произойдёт, если два одинаковых селектора с разным цветом идут подряд?
4. Почему лучше использовать классы, а не id для стилизации?
5. Как “реанимировать” стиль, если всё перебито библиотекой?