**CSS-переходы и анимации**

**9.1. Что такое CSS-переходы и зачем они нужны?**

CSS-переходы (transitions) позволяют плавно изменять CSS-свойства элементов при изменении их состояний. Это добавляет сайту динамичности, интерактивности и приятного UX.

**Преимущества переходов и анимаций:**

* Улучшение пользовательского опыта
* Плавность взаимодействий
* Привлекательность интерфейса
* Отсутствие необходимости JavaScript для базовых анимаций

**9.2. Основы CSS-переходов (transition)**

**Базовый синтаксис:**

css

КопироватьРедактировать

.element {

transition: свойство длительность timing-функция задержка;

}

**Пример: плавное изменение цвета кнопки**

css

КопироватьРедактировать

.button {

background-color: blue;

transition: background-color 0.3s ease-in-out;

}

.button:hover {

background-color: red;

}

**9.3. Управление свойствами переходов**

**Свойства transition:**

* **transition-property** (какое свойство изменяется)
* **transition-duration** (длительность перехода)
* **transition-timing-function** (характер изменения)
* **transition-delay** (задержка перед стартом)

Пример с указанием всех свойств отдельно:

css

КопироватьРедактировать

.button {

transition-property: background-color, color;

transition-duration: 0.3s, 0.5s;

transition-timing-function: ease-in, ease-out;

transition-delay: 0s, 0.2s;

}

**Сокращённая запись:**

css

КопироватьРедактировать

.button {

transition: background-color 0.3s ease-in, color 0.5s ease-out 0.2s;

}

**9.4. CSS transform — трансформация элементов**

Позволяет перемещать, вращать, масштабировать и искажать элементы.

**Основные функции transform:**

* translate() — перемещение по осям
* rotate() — вращение
* scale() — масштабирование
* skew() — искажение (наклон)

**Пример кнопки с увеличением при наведении:**

css

КопироватьРедактировать

.button {

transition: transform 0.3s ease;

}

.button:hover {

transform: scale(1.1);

}

**9.5. CSS keyframes — сложные анимации**

Когда нужны более сложные и многоэтапные анимации, используются ключевые кадры (keyframes).

**Пример простейшей keyframe-анимации:**

css

КопироватьРедактировать

@keyframes fadeIn {

from { opacity: 0; }

to { opacity: 1; }

}

.element {

animation: fadeIn 1s ease-in;

}

**Сложная анимация с несколькими кадрами:**

css

КопироватьРедактировать

@keyframes moveAndRotate {

0% { transform: translateX(0); }

50% { transform: translateX(100px) rotate(180deg); }

100% { transform: translateX(0) rotate(360deg); }

}

.box {

animation: moveAndRotate 3s infinite ease-in-out;

}

**9.6. Свойства анимаций**

* **animation-name** (название keyframes)
* **animation-duration** (длительность)
* **animation-timing-function** (функция анимации)
* **animation-delay** (задержка перед началом)
* **animation-iteration-count** (количество повторов)
* **animation-direction** (направление)
* **animation-fill-mode** (состояние после завершения)

**Сокращённая запись:**

css

КопироватьРедактировать

.element {

animation: fadeIn 2s ease-in 0.5s infinite alternate;

}

**9.7. Современные возможности анимаций**

**CSS-фильтры и анимации**

css

КопироватьРедактировать

.image:hover {

filter: grayscale(0);

}

.image {

filter: grayscale(100%);

transition: filter 0.4s ease;

}

**SVG-анимации через CSS**

SVG-элементы тоже отлично анимируются через CSS:

css

КопироватьРедактировать

.icon {

fill: black;

transition: fill 0.3s;

}

.icon:hover {

fill: red;

}

**9.8. Лайфхаки и советы по работе с переходами и анимациями**

* Используй плавные и естественные временные функции (ease-in-out).
* Не перегружай страницу слишком большим количеством анимаций.
* Используй минимальные времена переходов (0.2–0.5s) для комфортного восприятия.
* Старайся не анимировать layout-свойства (ширина, высота) напрямую — это ресурсоёмко.

**9.9. Лучшие практики CSS-анимаций**

* Анимируй opacity и transform для лучшей производительности.
* Минимизируй количество одновременно идущих анимаций.
* Используй will-change для оптимизации производительности браузеров:

css

КопироватьРедактировать

.button {

will-change: transform;

}

**9.10. Частые ошибки и анти-паттерны CSS-анимаций**

* Слишком долгие или короткие переходы.
* Использование громоздких layout-свойств для анимаций (margin, width, height).
* Избыточное количество анимаций и переходов.

**9.11. Глоссарий по анимациям и переходам**

| **Термин** | **Определение** |
| --- | --- |
| Transition | Плавный переход свойства |
| Transform | Трансформация элемента |
| Keyframes | Ключевые кадры сложной анимации |
| Animation | Сложная анимация |
| Animation-duration | Длительность анимации |
| Timing-function | Временная функция перехода |
| Animation-delay | Задержка перед стартом |
| Iteration-count | Количество повторов анимации |
| Animation-fill-mode | Финальное состояние после анимации |

**9.12. Вопросы для самопроверки**

1. Чем отличаются CSS-переходы от CSS-анимаций?
2. Какие свойства лучше всего подходят для плавных переходов?
3. Как создать сложную многоступенчатую анимацию?
4. Почему важно использовать will-change?
5. Как избежать проблем с производительностью при использовании анимаций?