HTML

Frontend Shortcuts

25 полезных сниппетов HTML, CSS, JS для ускорения разработки

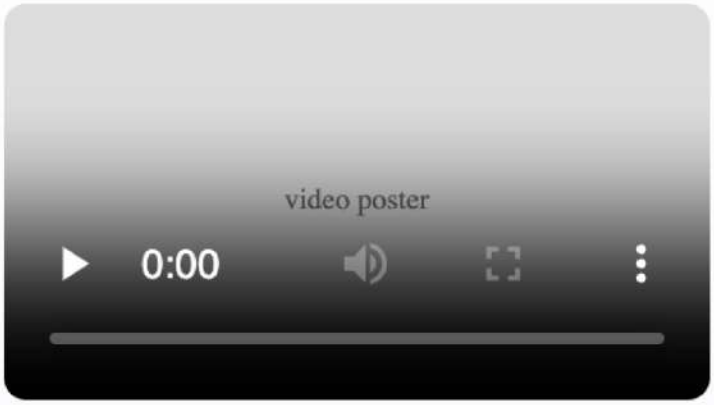
CSS

1. Lazy-loading изображений и видео

```
<img src="img.jpg" loading="lazy" alt="..." width="600" height="400">
<video controls preload="none" poster="preview.jpg">
  <source src="video.mp4" type="video/mp4">
</video>
```

1) Lazy-loading изображений и видео

Атрибут `loading="lazy"` и `preload="none"`.



2. Автоматическая генерация ссылок для телефона и email

```
<a href="tel:+79991234567">Позвонить</a>
<a href="mailto:mail@example.com">Написать</a>
```

2) Ссылки tel / mail

Позвонить +7 999 123-45-67 • Написать

3. `<details>` и `<summary>` для спойлеров без JS

```
<details>
  <summary>Подробнее</summary>
  <p>Скрытый текст</p>
</details>
```

3) <details> <summary> (спойлер без JS)

▶ Нажми, чтобы раскрыть

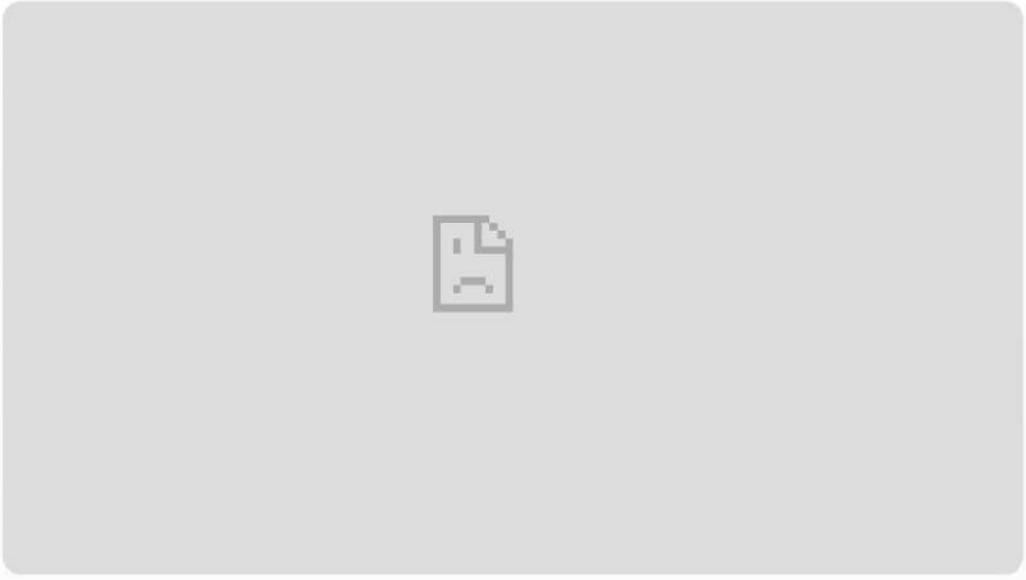
4. Использование `<template>` для динамической вставки

```
<template id="card-template">
  <div class="card">
    <h3></h3>
    <p></p>
  </div>
</template>
```

5. Геолокация в `<iframe>` Google Maps

```
<iframe
  src="https://www.google.com/maps/embed?pb=!1m18..."
  width="600" height="450" style="border:0;" loading="lazy">
</iframe>
```

5) Встраиваемая карта (iframe)

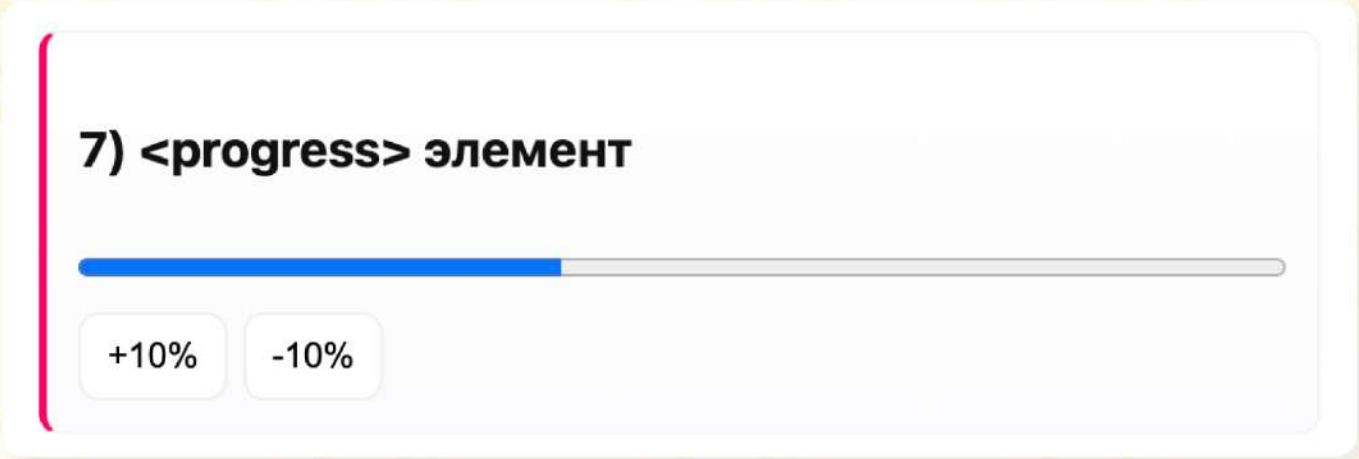


6. Meta viewport с запретом масштабирования

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

7. `<progress>` для отображения прогресса без JS-анимаций

```
<progress value="70" max="100">70%</progress>
```



8. Семантическая разметка статьи

```
<article>
  <header><h1>Заголовок</h1></header>
  <section>Содержимое</section>
  <footer>Автор</footer>
</article>
```

9. CSS-анимация появления

```css
@keyframes fadeIn {
  from { opacity: 0; }
  to { opacity: 1; }
}
.fade-in { animation: fadeIn 0.5s ease-in; }
```

10. CSS Grid для галереи

```css
.gallery {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 10px;
}
```

11. Темная тема через CSS-переменные

```css
:root {
  --bg: white;
  --text: black;
}
@media (prefers-color-scheme: dark) {
  :root {
    --bg: black;
    --text: white;
  }
}
body {
  background: var(--bg);
  color: var(--text);
}
```

12. CSS для адаптивного iframe

```css
.iframe-wrapper {
  position: relative;
  padding-bottom: 56.25%;
  height: 0;
}
.iframe-wrapper iframe {
  position: absolute;
  width: 100%;
  height: 100%;
}
```

13. Глитч-эффект текста

```css
.glitch {
  position: relative;
  color: white;
}
.glitch::before,
.glitch::after {
  content: attr(data-text);
  position: absolute;
  left: 0;
}
.glitch::before {
  animation: glitch 1s infinite;
  color: red;
}
.glitch::after {
  animation: glitch 1s infinite reverse;
  color: blue;
}
```

14. CSS-анимация скелетон-лоадера

```css
.skeleton {
  background: linear-gradient(90deg, #eee, #f5f5f5, #eee);
  background-size: 200% 100%;
  animation: skeleton 1.5s infinite linear;
}
@keyframes skeleton {
  from { background-position: 200% 0; }
  to { background-position: -200% 0; }
}
```

15. CSS clip-path для нестандартных форм

```css
.circle {
  clip-path: circle(50% at 50% 50%);
}
```

16. Плавное появление элементов при скролле (без JS)

```css
[data-animate] {
  opacity: 0;
  transition: opacity 0.6s ease;
}
[data-animate].visible {
  opacity: 1;
}
```

17. CSS backdrop-filter (стеклянный эффект)

```css
.glass {
  backdrop-filter: blur(10px);
  background: rgba(255, 255, 255, 0.2);
}
```

9–17) CSS: grid, fade, theme vars, iframe wrapper, glitch, skeleton, clip-path, scroll reveal, glass

Gallery (Grid)





Glitch

GLITCH

Skeleton loader

Загрузить

Clip-path avatar

18. Динамическая загрузка шаблона из `<template>`

```
const tmpl = document.querySelector('#card-template');
const clone = tmpl.content.cloneNode(true);
clone.querySelector('h3').textContent = 'Заголовок';
document.body.appendChild(clone);
```

19. Отслеживание видимости элемента через IntersectionObserver

```
const observer = new IntersectionObserver(entries => {
  entries.forEach(e => e.target.classList.toggle('visible', e.isIntersecting));
});
document.querySelectorAll('[data-animate]').forEach(el => observer.observe(el));
```

20. Асинхронный импорт JS-модуля

```
import('module.js').then(mod => mod.init());
```

21. Загрузка JSON с кэшем в localStorage

```
async function getData(url) {
  const cache = localStorage.getItem(url);
  if (cache) return JSON.parse(cache);
  const data = await fetch(url).then(r => r.json());
  localStorage.setItem(url, JSON.stringify(data));
  return data;
}
```

22. Клонирование объекта без мутаций

```
const copy = structuredClone(original);

function deepClone(data) {
  if (typeof structuredClone === "function") {
    return structuredClone(data);
  } else {
    return JSON.parse(JSON.stringify(data));
  }
}
```

23. Генерация уникального ID

```
const uid = crypto.randomUUID();
```

24. Throttle для оптимизации скролла

```
function throttle(fn, limit) {
  let inThrottle;
  return function(...args) {
    if (!inThrottle) {
      fn(...args);
      inThrottle = true;
      setTimeout(() => inThrottle = false, limit);
    }
  };
}
```

25. Отслеживание сети (онлайн/офлайн)

```
window.addEventListener('online', () => console.log('Онлайн'));
window.addEventListener('offline', () => console.log('Офлайн'));
```

18–25) JS примеры

Добавить из <template>

Динамически импортировать модуль

Получить данные (с кэшем)

```
Результат...
```

Клонировать объект

```
original → copy
```

Сгенерировать UUID

Копировать текст

Состояние сети: **online**