ПМ3 Разработка модулей ПО.

РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

Тема 4. Регулярные выражения.

Лекция 22. Жадные и ленивые совпадения.

Цель занятия:

Познакомиться регулярными выражениями в JavaScript как инструментом для поиска и обработки текста.

Учебные вопросы:

1. Жадные и ленивые совпадения.

2. Позиционные проверки (Lookahead / Lookbehind).

1. Жадные и ленивые совпадения.

- Квантификаторы по умолчанию — жадные

Квантификаторы (+, *, ?, {n,m}) по умолчанию жадные — они захватывают как можно больше символов, сохраняя при этом соответствие шаблону.

Пример:

```
const text = "<b>Жирный текст</b> и <i>курсив</i>";
console.log(text.match(/<.+>/));
// ["<b>Жирный текст</b> и <i>курсив</i>"]
```

◇ Ленивые совпадения — *?, +?, ??, {n,m}?

Если после квантификатора добавить ?, он станет ленивым (lazy) — будет брать минимум символов, достаточный для совпадения.

Пример:

```
const text = "<b>Жирный текст</b> и <i>курсив</i>";
console.log(text.match(/<.+?>/g));
// ["<b>", "</b>", "<i>", "</i>"]
```

Сравнение

Квантификатор	Жадный (по умолчанию)	Ленивый (с ?)
*	.* — захватит всё	.*? — захватит по минимуму
+	.+ — минимум 1, максимум всё	.+? — минимум 1, но как можно меньше
?	.? — 0 или 1, жадно	?? — 0 или 1, лениво
{n,m}	{n,m} — максимально m	{n,m}? — минимально n

◇ Практические примеры

1. HTML-теги

```
"<p>Hello</p><p>World</p>".match(/<.+>/)
// ["<p>Hello</p><p>World</p>"]

"<p>Hello</p><p>World</p>".match(/<.+?>/g)
// ["<p>", "</p>", "<p>", "</p>"]
```

2. Выделение текста в кавычках

```
const text = '"JS" и "Regex"';
console.log(text.match(/".+"/));
// ["\"JS\" и \"Regex\""]

console.log(text.match(/".+?"/g));
// ["\"JS\"", "\"Regex\""]
```

3. Поиск цен

```
const text = "350000тг, 5000тг, 1500тг";
console.log(text.match(/\d+тг/g));
// ["350000тг", "5000тг", "1500тг"]
```

(здесь жадность не мешает, потому что есть ограничитель — тг).

📌 Итоги

- По умолчанию квантификаторы жадные.
- Добавление ? делает их ленивыми.
- Жадные нужны, когда надо «схватить всё», ленивые — когда важны ограничители (например, HTML-теги, кавычки).

2. Позиционные проверки Lookahead/Lookbehind.

Lookahead и Lookbehind — это проверки в регулярных выражениях, которые смотрят «вперёд» или «назад» от текущей позиции, не включая найденное в результат.

Их ещё называют zero-width assertions — проверки нулевой ширины: они не захватывают текст, а лишь проверяют условие.

Lookahead — «смотрим вперёд»

X(?=Y) — найти X, если сразу после него идёт Y

X(?!Y) — найти X, если сразу после него НЕ идёт Y

Примеры:

```
// Найти числа, за которыми стоит $
"350тг 120$ 5000$".match(/\d+(?=\$)/g)
// ["120", "5000"]

// Найти числа, за которыми НЕТ $
"350тг 120$ 5000$".match(/\d+(?!\d*\$)/g)
// ["350"]
```

Lookbehind — «смотрим назад»

(?<=Y)X — найти X, если перед ним есть Y

(?<!Y)X — найти X, если перед ним НЕТ Y

Примеры:

```
// Найти домен e-mail (всё, что идёт после @)
"student@mail.com".match(/(?<=@)\w+\.\w+/)
// ["mail.com"]

// Найти числа, перед которыми нет знака $
"350тг $120".match(/(?<!\$)\b\d+/g)
// ['350']
```

◇ Сравнение

Проверка	Синтаксис	Что ищет
Позитивный lookahead	X(?=Y)	X, перед которым есть Y
Негативный lookahead	X(?!Y)	X, перед которым нет Y
Позитивный lookbehind	(?<=Y)X	X, за которым идёт Y
Негативный lookbehind	(?<!Y)X	X, за которым не идёт Y

◇ Практические примеры

Email: отделить логин от домена

```
const email = "student@mail.com";
console.log(email.match(/^[^@]+(?=@)/)[0]); // "student"
console.log(email.match(/(?<=@).+$/)[0]);   // "mail.com"
```

Телефон: выделить код оператора

```
const phone = "+7 (701) 123-45-67";
console.log(phone.match(/(?<=\()\d+(?=\))/)[0]); // "701"
```

Цены: найти только в долларах

```
const prices = "350000тг, 5000$, 1500тг, 20$";
console.log(prices.match(/\d+(?=\$)/g)); // ["5000", "20"]
```

📌 Итоги:

Lookahead = смотрим вперёд (?=... или ?!...).

Lookbehind = смотрим назад (?<=... или ?<!...).

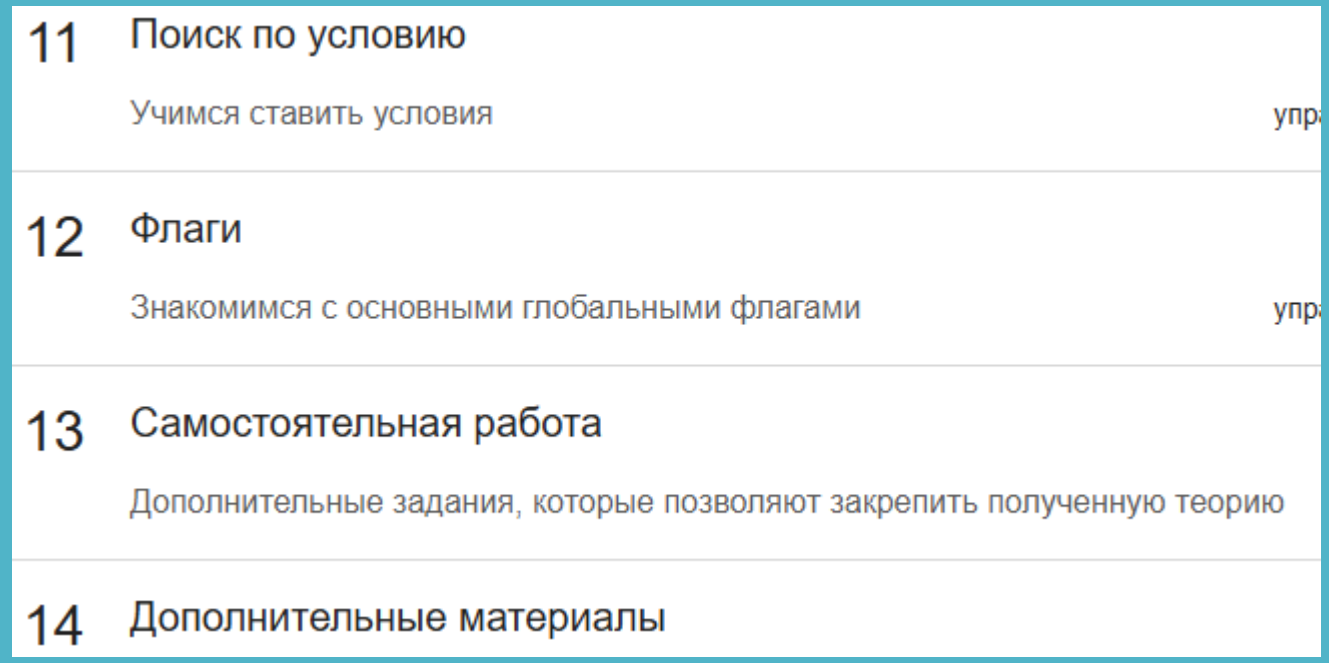
Эти проверки **ничего не добавляют в результат**, они просто ограничивают, что можно взять.

Контрольные вопросы:

- Чем отличается жадный квантификатор от ленивого?
- Что делает *?, +?, {n,m}??
- Почему в строке "hiok" выражение /<.+>/ находит всё сразу?
- Как изменить выражение, чтобы найти каждый тег по отдельности?
- Что такое lookahead и lookbehind? Чем они отличаются от обычных групп?
- Объясните разницу между (?=...) и (?!...).
- Объясните разницу между (?<=...) и (?<!...).
- Как с помощью lookahead найти все цены только в долларах в строке "350000тг, 5000$, 1500тг, 20$"?
- Как с помощью lookbehind извлечь домен из e-mail "student@mail.com"?
- Что означает (?i) внутри регулярки?
- Можно ли локально выключить флаг? Как?
- Как работает конструкция (?(1)yes|no)?
- Как проверить, что пароль содержит хотя бы одну букву и хотя бы одну цифру?
- Чем условные проверки отличаются от обычных групп?

Домашнее задание:

1. https://ru.hexlet.io/courses/regexp



2. Повторить материалы лекции.

Материалы лекций:

https://github.com/ShViktor72/Education2025

Обратная связь:

colledge20education23@gmail.com