1. Виды условных операторов в JS:

- if (с одной ветвью);

- if...else (с двумя ветвями);

- else if... (с несколькими ветвями);

- условный тернарный оператор ?:

- инструкция выбора switch

1. let date = new Date();

let time = date.getHours();

time < 10 ? alert( "Доброе утро!" ) : alert( "Добрый день!" );

1. Выведется (но я не очень понимаю, что значит условие if”0”- что именно тут false? Проверила, если поставить if”1” – “true” – тоже все выведется. Буду благодарна за пояснение))
2. 4.1 true
   1. false
   2. true
3. три способа назначения обработчиков событий:

* Атрибут HTML: onclick="..." .
* DOM-свойство: elem. onclick = function .
* Специальные методы: elem. addEventListener(event, handler[, phase]) для добавления, removeEventListener для удаления.

Самый универсальный – addEventListener (т.к. с помощью него можно навесить несколько обработчиков на одно событие. Также, обработчики некоторых событий можно назначать только через `addEventListener`: существуют события, которые нельзя назначить через DOM-свойство, но можно через `addEventListener`, например, событие `DOMContentLoaded`, которое срабатывает, когда завершена загрузка и построение DOM документа)

1. Корректна. Пример:

button.onclick = function() {

sayThanks(); // содержимое атрибута

};

1. События клавиатуры:

- keydown : возникает при нажатии клавиши клавиатуры и длится, пока нажата клавиша

- keyup : возникает при отпускании клавиши клавиатуры

- keypress : возникает при нажатии клавиши клавиатуры, но после события keydown и до события keyup.

Нашла доп. Инфу по вопросу:

Для работы с событиями клавиатуры определен объект KeyboardEvent, который добавляет к свойствам объекта Event ряд специфичных для клавиатуры свойств:

* altKey: возвращает true, если была нажата клавиша Alt во время генерации события
* key: возвращает символ нажатой клавиши, например, при нажатии на клавишу «T» это свойство будет содержать «T». А если нажата клавиша «Я», то это свойство будет содержать «Я»
* code: возвращает строковое представление нажатой клавиши физической клавиатуры QWERTY, например, при нажатии на клавишу «T» это свойство будет содержать «KeyT», а при нажатии на клавишу «;» (точка запятой), то свойство возвратит «Semicolon».

При использовании этого свойства следует учитывать ряд момент. Прежде всего используется клавиатура QWERTY. То есть мы переключим раскладку, к примеру, на русскоязычную и нажмем на клавишу «Я», то значением будет «KeyZ» – на клавиатуре QWERTY клавиша Z представляет ту же клавишу, что и на русскоязычной раскладке «Я»

Другой момент – учитывается именно физическая клавитура. Если нажата клавиша на виртуальной клавиатуре, то возвращаемое значение будет устанавливаться браузером исходя из того, какой клавише на физической клавиатуре соответствовало нажатие.

* ctrlKey: возвращает true, если была нажата клавиша Ctrl во время генерации события
* metaKey: возвращает true, если была нажата во время генерации события метаклавиша клавиатуры
* shiftKey: возвращает true, если была нажата клавиша Shift во время генерации события

1. Объект события: чтобы хорошо обработать событие, могут понадобиться детали того, что произошло. Не просто «клик» или «нажатие клавиши», а также – какие координаты указателя мыши, какая клавиша нажата и так далее.

Когда происходит событие, браузер создаёт **объект события**, записывает в него детали и передаёт его в *качестве аргумента* функции-обработчику.

Пример ниже демонстрирует получение координат мыши из объекта события:

<input type="button" value="Нажми меня" id="elem">

<script>

elem.onclick = function(event) {

// вывести тип события, элемент и координаты клика

alert(event.type + " на " + event.currentTarget);

alert("Координаты: " + event.clientX + ":" + event.clientY);

};

</script>

Некоторые свойства объекта event:

event.type

Тип события, в данном случае "click".

event.currentTarget

Элемент, на котором сработал обработчик. Значение – обычно такое же, как и у this, но если обработчик является функцией-стрелкой или при помощи bind привязан другой объект в качестве this, то мы можем получить элемент из event.currentTarget.

event.clientX / event.clientY

Координаты курсора в момент клика относительно окна, для событий мыши.

Вывод: не важно, как мы назначаем обработчик – он получает объект события первым аргументом. Этот объект содержит подробности о том, что произошло.

Каждое событие может содержать дополнительные свойства в зависимости от его типа, но список ниже есть у всех:

* bubbles – является ли данное событие всплывающим.
* cancelable – является ли событие отменяемым.
* currentTarget – указывает на элемент, на котором установлен обработчик события.
* defaultPrevented – отменено ли поведение события по умолчанию.
* eventPhase – указывает на фазу срабатывания события.
* isTrusted – указывает на происхождение события, будет в значении true, если событие инициировано действиями пользователя. false - если событие инициировано из кода с помощью dispatchEvent().
* target – ссылка на объект, которым было инициировано событие. Например, если событие произошло на поле ввода, мы получим ссылку на этот DOM элемент.
* timeStamp – время возникновения события в миллисекундах.
* type – тип события.

1. Операторы сравнения:

Больше/меньше: a > b, a < b.

Больше/меньше или равно: a >= b, a <= b.

Равно: a == b

Важно: для сравнения используется двойной знак равенства ==. Один знак равенства a = b означал бы присваивание.

Не равно. В математике обозначается символом ≠, но в JavaScript записывается как a != b.