skillup

Event

План

- Події в браузері
- Порядок обробки подій
- Об’єкт події
- Вспливання подій
- Перехват події
- click
- scroll
- types
- події форм

Події (event)

- Для реакції на дії користувача і внутрішньої взаємодії скриптів існують події
- Подія - це сигнал від браузера про те, що щось сталося
- Події можна призначити обробник, тобто функцію, яка спрацює, як тільки подія відбулася

https://learn.javascript.ru/introduction-browser-events

Задання обробників подій

- через атрибут

```
<input value="Click me" onclick="alert('Клик!')" type="button">
<input value="Click me" onclick="functionEvent()" type="button">
```

- через властивість DOM-елемента

```
<input id="elem" type="button" value="Click me" />
<script>
  elem.onclick = function() {
    alert( 'Ok' );
  };
</script>
```

- Методи `addEventListener` і `removeEventListener` є сучасним способом призначити або видалити обробник

```
elem.addEventListener( "click" , function() {alert('Ok!')});
// ....
elem.removeEventListener( "click", function() {alert('Ok!')});
```

```
function handler() {
  alert( 'Ok!' );
}
input.addEventListener("click", handler);
```

this в подіях

- Усередині обробника події this посилається на поточний елемент, тобто на той, на якому він спрацював
- Це можна використовувати, щоб отримати властивості або змінити елемент

```
<button onclick="alert(this.innerHTML)">Click me</button>
```

Практика

<input type=”text”><button id=”btn”>Click</button>

- додати обробник на кнопку, при кліку на яку буде в консоль виводитися текст з поля

Види подій

Події миші:

click - відбувається, коли склікали на елемент лівою кнопкою миші

dblclick - відбувається, коли двічі склікали на елемент лівою кнопкою миші

contextmenu - відбувається, коли склікали на елемент правою кнопкою миші

mouseover - виникає, коли на елемент наводиться миша

mousedown і **mouseup** - коли кнопку миші натиснули або віджали

mousemove - при русі миші

https://developer.mozilla.org/en-US/docs/Web/Events

Події на елементах управління:

submit - відвідувач відправив форму <form>

focus - відвідувач фокусується на елементі, наприклад натискає на <input>

Клавіатурні події:

keydown - коли відвідувач натискає клавішу

keyup - коли відвідувач відпускає клавішу

Події документа:

DOMContentLoaded - коли HTML завантажений і оброблений, DOM документа повністю побудований і доступний.

https://learn.javascript.ru/introduction-browser-events

об’єкт події

```
elem.addEventListener( "click" , function(event) {
    console.log(event)
});
```

- виведе всі властивості об’єкта події

https://learn.javascript.ru/obtaining-event-object

Практика

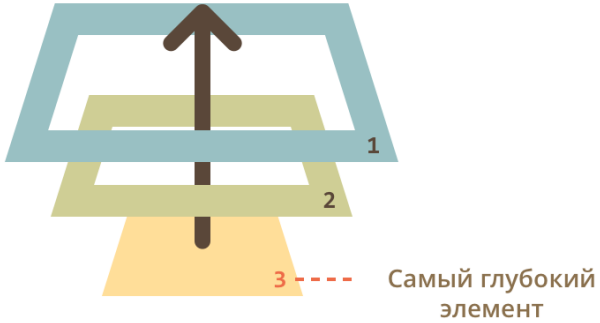
- написати обробник на подію клік щоб елемент переміщувався на координати кліку
- зробити dropdown(випадаючий список)

Вспливання подій (bubbling)

- При настанні події обробники спочатку спрацьовують на самому вкладеному елементі, потім на його батьку, потім вище і так далі, вгору по ланцюжку вкладеності

https://jsfiddle.net/blog_code/481sya2e/

https://learn.javascript.ru/event-bubbling



Цільовий елемент event.target

- Найглибший елемент, який викликає подія, називається «цільовим» або «вихідним» елементом і доступний як event.target.
- event.target - це вихідний елемент, на якому відбулася подія, в процесі спливання він незмінний.
- this(=event.currentTarget) - це поточний елемент, до якого дійшло спливання, на ньому зараз виконується оброблювач.

http://plnkr.co/edit/i7dmLKNVcIzcbqRb6h9o?p=preview

Для зупинки спливання потрібно викликати метод **event.stopPropagation()**.

Делегування подій

- якщо у нас є багато елементів, події на яких потрібно обробляти схожим чином, то замість того, щоб призначати обробник кожному - ми ставимо один обробник на їх загального предка

https://jsfiddle.net/blog_code/2ns610yw/8/

https://learn.javascript.ru/event-delegation

Скасування дії браузера

- Для скасування дії браузера існує стандартний метод **event.preventDefault()**.

```
elem.addEventListener( "click" , function(event) {
        event.preventDefault();
    console.log(event);
});
```

Події мишки

- **mousedown** кнопка миші натиснута над елементом
- **mouseup** кнопка миші відпущена над елементом
- **mouseover** миша з'явилася над елементом
- **mouseout** миша пішла з елемента
- **mousemove** кожен рух миші над елементом генерує ця подія
- **click** викликається при кліці мишею, тобто при mousedown, а потім mouseup на одному елементі
- **contextmenu** викликається при натисканні правою кнопкою миші на елементі.
- **dblclick** викликається при подвійному натисканні по елементу.

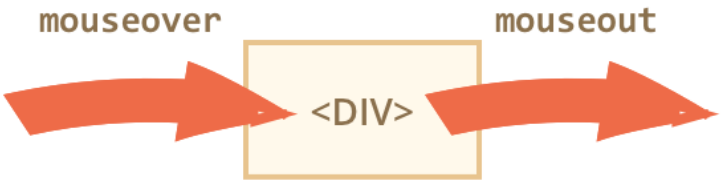
Комбінація кліку

- shiftKey
- altKey
- ctrlKey
- metaKey (для Mac)

```
elem.addEventListener( "click" , function(event) {
    console.log(event.altKey);
});
```

Рух мишки

- Подія mouseover відбувається, коли миша з'являється над елементом, а mouseout - коли йде з нього



- При цьому ми можемо дізнатися, з якого елементу прийшла (або на який пішла) миша, використовуючи додаткове властивість об'єкта події relatedTarget
- Події mousemove і mouseover / mouseout спрацьовують так часто, наскільки це дозволяє внутрішня система взаємодії з мишею браузера

Практика

- підсвічувати комірки таблиці при наведенні

Scroll

- При прокручуванні спрацьовує подія **onscroll**. Воно відбудеться за будь-якої прокручуванні, в тому числі через клавіатуру, але тільки на прокручуваних елементах. Наприклад, елемент з **overflow: hidden** в принципі не може згенерувати onscroll.
- А подія **wheel** є чисто «мишачим». Воно генерується над будь-яким елементом при пересуванні колеса миші. При цьому не важливо, прокручуваний він чи ні. Зокрема, **overflow: hidden** ніяк не перешкоджає обробці колеса миші.

Подія scroll

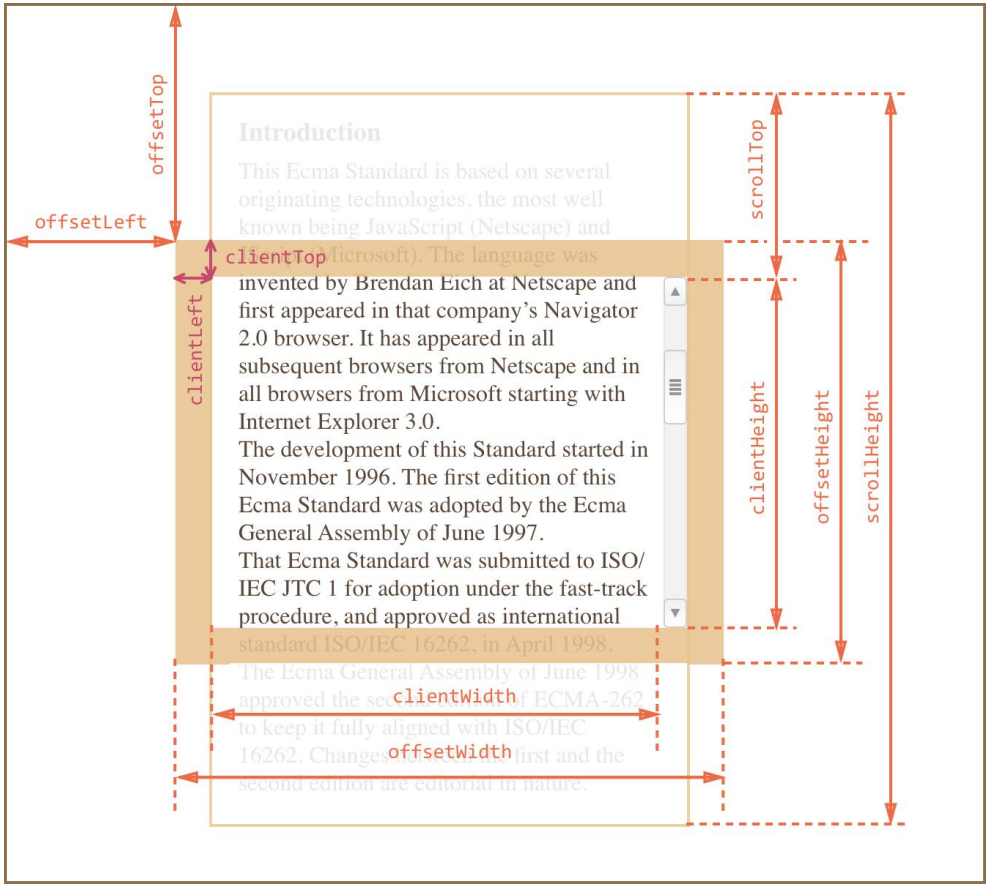
- scrollTop виводиться кількість пікселів прокручених від верхньої межі

```
window.addEventListener('scroll', function(e) {
    console.log(this.scrollY);
});
```

https://jsfiddle.net/blog_code/v3naLtj9/4/

https://learn.javascript.ru/onscroll

Розміри елементів



- координати рахуються з верхнього лівого кута
- **offsetLeft / offsetTop** - позиція в пікселях лівого верхнього кута блоку
- **offsetWidth / offsetHeight** - «зовнішня» ширина / висота блоку, включаючи рамки
- **clientLeft / clientTop** - відступ області вмісту від лівого-верхнього кута елемента. Якщо операційна система має вертикальну прокрутку праворуч, то рівні ширинам лівої / верхньої рамки, якщо ж зліва (ОС на івриті, арабською), то clientLeft включає в себе прокрутку
- **clientWidth / clientHeight** - ширина / висота вмісту разом з полями padding, але без смуги прокрутки
- **scrollWidth / scrollHeight** - ширина / висота вмісту, включаючи прокручуємо область. Включає в себе padding і не включає смуги прокрутки
- **scrollLeft / scrollTop** - ширина / висота прокрученої частини документа, вважається від верхнього лівого кута

https://learn.javascript.ru/metrics

Практика

- при прокручуванні сторінки на 200px тоглити клас в header

Події вікна

- зміна розмірів вікна

```
window.addEventListener("resize", function(){});
```

- зміна орієнтації екрану (для мобільних дивайсів)

```
window.addEventListener("orientationchange", function(){});
```

Розміри вікна

- screen.width, screen.height - повертає довжину і ширину екрану дивайса

```
window.addEventListener('resize', function() {
        console.log(this.innerWidth); // розмір екрана
});
```

Практика

- зробити перевірку на ширину екрану, якщо менше 768px то до елемента додати клас

Події клавіатури

- Події **keydown** / **keyup** відбуваються при натисканні / відпуску клавіші і дозволяють отримати її скан-код у властивості keyCode

  https://learn.javascript.ru/keyboard-events

Приклад

```
var number = document.querySelector('.number');

number.addEventListener('keypress', function(e) {



});
```

Практика

- зробити інтуп (type=”text”) в який можна вводити тільки цифри (keyCode для цисел від 48 до 57 і від 96 до 105)

https://codepen.io/jurj-shewchuk/pen/pGbeEY

Події загрузки документа

- DOMContentLoaded - означає, що все DOM-елементи розмітки вже створені, можна їх шукати, вішати обробники, створювати інтерфейс, але при цьому, можливо, ще не довантажити якісь картинки або стилі.
- load - сторінка і все ресурси завантажені, використовується рідко, зазвичай немає потреби чекати цього моменту.
- beforeunload / unload - можна перевірити, чи зберіг відвідувач зміни, уточнити, чи дійсно він хоче покинути сторінку.

https://learn.javascript.ru/onload-ondomcontentloaded

Навігація по формах

```
<form name=”form”>
  <input type="text" name="surname">
</form>
<script>
var form = document.forms[name/index];
var elem = form.elements[name/index];
</script>
```

- input значення ставиться / читається через властивість value

Події у формах

Події **focus** / **blur**

- Подія focus викликається тоді, коли користувач фокусується на елементі
- blur - коли фокус зникає, наприклад відвідувач клацає на іншому місці екрана

https://learn.javascript.ru/events-change

Подія **change**

- Подія change відбувається після закінчення зміни значення елемента форми, щоб ці поправки зафіксовано
- Для текстових елементів це означає, що подія відбудеться не при кожному введенні, а при втраті фокуса

Подія **input**

- Подія input спрацьовує тут же при зміні значення текстового елемента і підтримується всіма браузерами, крім IE8-

Подія і метод submit

Щоб відправити форму на сервер, у відвідувача є два способи:

- Перший - це натиснути кнопку <input type = "submit"> або <input type = "image">
- Другий - натиснути Enter, перебуваючи на якомусь полі
- Щоб відправити форму на сервер з JavaScript - потрібно викликати на елементі форми метод form.submit()

https://learn.javascript.ru/forms-submit

Посилання

https://www.w3schools.com/js/js_events.asp
https://www.w3schools.com/js/js_events_examples.asp
https://www.w3schools.com/js/js_htmldom_eventlistener.asp
https://www.w3schools.com/js/js_htmldom_events.asp
https://learn.javascript.ru/events-and-interfaces
https://learn.javascript.ru/document

Відео

https://www.youtube.com/watch?v=oR1tCXv9pVU