

# Интерфейсы ввода

Лекция 7

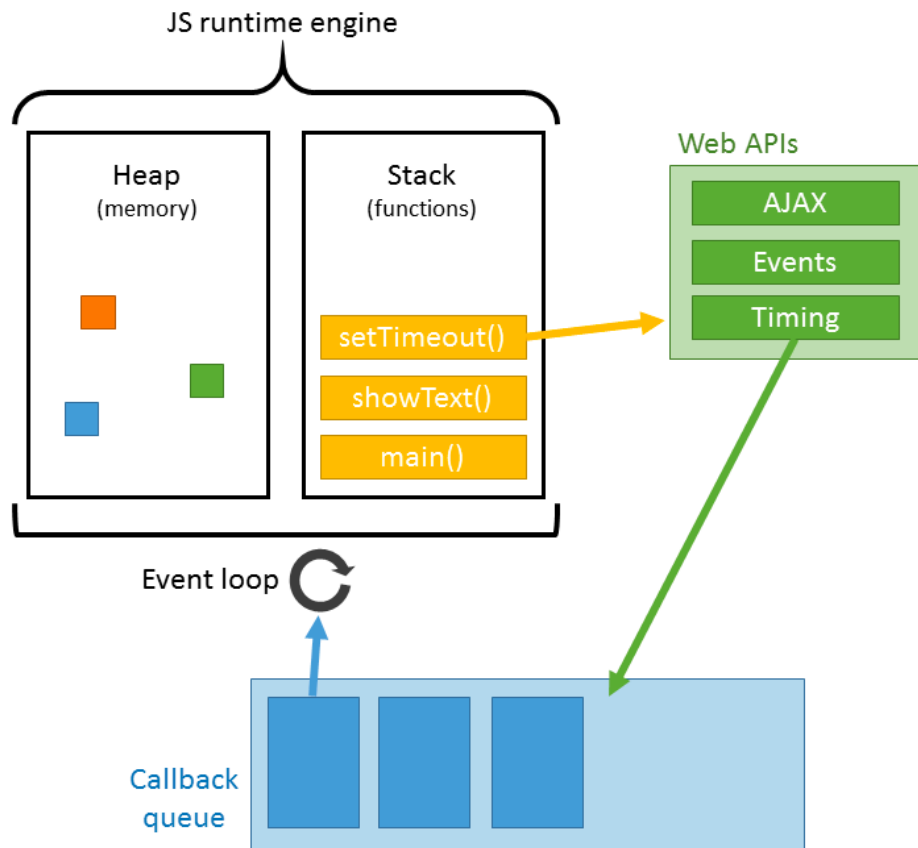
Михаил Привер

- Отметиться на портале
- Оставить обратную связь

- Event Loop
- События DOM
- File API
- Geolocation API
- Drag & Drop
- Audio API
- Media Devices API
- Vibration API
- Payment API

## Запуск функции создает контекст выполнения

```
1. function foo(b) {  
2.     var a = 10;  
3.     return a + b + 11;  
4. }  
5.  
6. function bar(x) {  
7.     var y = 3;  
8.     return foo(x * y);  
9. }  
10.  
11. console.log(bar(7)); // вернет 42
```



```
1.while (eventLoop.waitForTask()) {  
2.    eventLoop.processNextTask();  
3.}
```

- Манипуляция с DOM
- Взаимодействие с пользователем
- Сетевое взаимодействие
- Переход между страницами (история)

```
1. while (eventLoop.waitForTask()) {  
2.     const taskQueue = eventLoop.selectTaskQueue();  
3.     if (taskQueue.hasNextTask()) {  
4.         taskQueue.processNextTask();  
5.     }  
6. }
```



```
1. while (eventLoop.waitForTask()) {  
2.     const taskQueue = eventLoop.selectTaskQueue();  
3.     if (taskQueue.hasNextTask()) {  
4.         taskQueue.processNextTask();  
5.     }  
6.  
7.     const microtaskQueue = eventLoop.microTaskQueue;  
8.     while (microtaskQueue.hasNextMicrotask()) {  
9.         microtaskQueue.processNextMicrotask();  
10.    }  
11.  
12.    if (eventLoop.shouldRender()) {  
13.        eventLoop.render();  
14.    }  
15.}
```

```
1. console.log('script start');
2.
3. setTimeout(function() {
4.     console.log('setTimeout');
5. }, 0);
6.
7. Promise.resolve().then(function() {
8.     console.log('promise1');
9. }).then(function() {
10.    console.log('promise2');
11. });
12.
13. console.log('script end');
```

<https://jakearchibald.com/2015/tasks-microtasks-queues-and-schedules/#why-this-happens>

## События мыши:

- **click** – происходит, когда кликнули на элемент левой кнопкой мыши (на устройствах с сенсорными экранами оно происходит при касании).
- **contextmenu** – происходит, когда кликнули на элемент правой кнопкой мыши.
- **mouseover** / **mouseout** – когда мышь наводится на / покидает элемент.
- **mousedown** / **mouseup** – когда нажали / отжали кнопку мыши на элементе.
- **mousemove** – при движении мыши.

## События на элементах управления:

- **submit** – пользователь отправил форму `<form>`.
- **focus** – пользователь фокусируется на элементе, например нажимает на `<input>`.

## Клавиатурные события:

- **keydown** и **keyup** – когда пользователь нажимает / отпускает клавишу.

## События документа:

- **DOMContentLoaded** – когда HTML загружен и обработан, DOM документа полностью построен и доступен.

## CSS events:

- **transitionend** – когда CSS-анимация завершена.

```
1. <input value="Нажми меня" onclick="alert('Клик!')" type="button">
```

```
1. <input id="elem" type="button" value="Нажми меня!">
2. <script>
3.     document.getElementById('elem').onclick = () => {
4.         alert('Спасибо');
5.     };
6. </script>
```

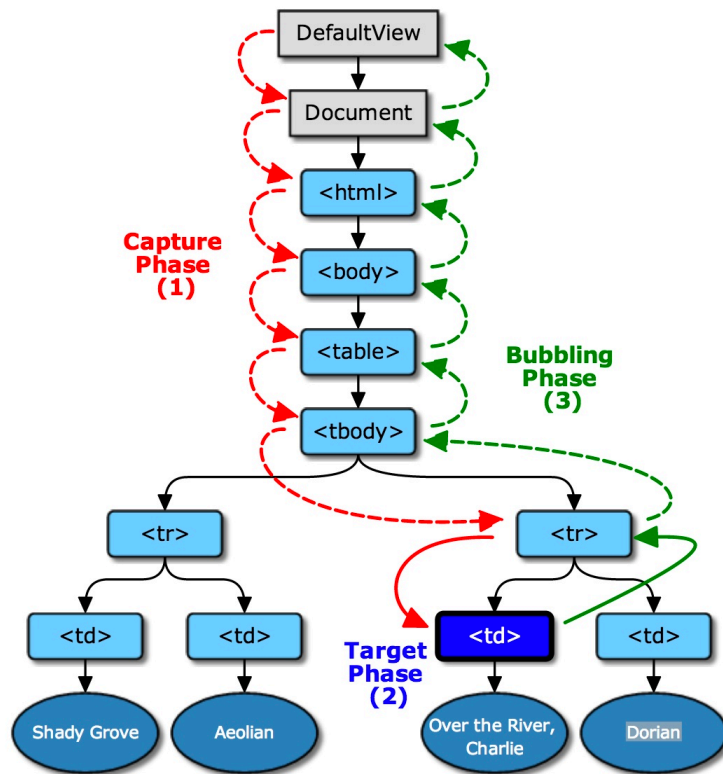
1. `target.addEventListener(type, listener[, useCapture]);`
2. `target.addEventListener(type, listener[, options]);`

- options
  - capture
  - once
  - passive
- Обработчики
  - вызываются в контексте target
  - с одинаковыми параметрами игнорируются
  - назначенные в момент обработки не будут выполнены

1. `target.removeEventListener(type, listener[, options]);`
2. `target.removeEventListener(type, listener[, useCapture]);`

- Опции и сигнатура те же
- Должны совпадать `type`, `listener` и опция `capture`

```
1. elem.addEventListener( "click" , () => alert('Спасибо!'));
2. // ...
3. // Не сработает!
4. elem.removeEventListener( "click", () => alert('Спасибо!'));
5.
6. function handler() {
7.   alert( 'Спасибо!' );
8. }
9.
10. input.addEventListener("click", handler);
11. // ...
12. // Сработает!
13. input.removeEventListener("click", handler);
```



- Три фазы
  - Захват
  - Обработка
  - Всплытие (не у всех)
- Имеют обработчик по умолчанию
- Обработчики назначаются на первую и третью фазы



## Использование в обработчиках

```
1. document.body.addEventListener(function (event) {
2.     event.type;                // тип
3.     event.eventPhase;          // фаза
4.     event.target;              // элемент захвативший событие
5.     event.currentTarget === this; // элемент, на котором происходит обработка
6.     event.bubbles               // всплытие
7.     event.cancelable            // отменяемость
8.     event.composed              // всплытие выше shadowRoot
9.     event.preventDefault();    // отмена действия по-умолчанию
10.    event.stopPropagation();    // отмена дальнейшего всплытия
11.    event.stopImmediatePropagation(); // отмена дальнейшей обработки
12. });
```

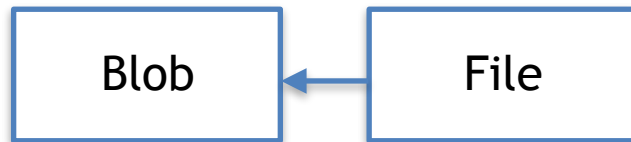
```
1. cancelled = !target.dispatchEvent(event)
```

## Принимает объект события

```
1. var event = new MouseEvent('click');  
2. elem.dispatchEvent(event);
```

```
1. var event = new CustomEvent('custom', {detail: 'data'});  
2. elem.addEventListener('custom', function (e) { ... }); // e === event  
3. elem.dispatchEvent(event);
```

- File.lastModified
- File.lastModifiedDate
- File.name
- File.size
- File.type
  
- Blob.text()
- Blob.arrayBuffer()
  
- FileList



```
1. <input type="file" id="input1">
2. <input type="file" multiple id="input2">
```

```
1. const selectedFile = document.getElementById('input1').files[0];
2.
3. const inputElement = document.getElementById('input2');
4.
5. inputElement.addEventListener('change', handleFiles, false);
6.
7. function handleFiles() {
8.     const fileList = this.files;
9.     // работаем со списком файлов
10. }
```

```
1. function updateSize() {  
2.     let bytes = 0;  
3.     let files = document.getElementById('input2').files;  
4.  
5.     for (let i = 0; i < files.length; i++) {  
6.         bytes += files[i].size;  
7.     }  
8.  
9.     document.getElementById('fileNum').innerHTML = files.length;  
10.    document.getElementById('fileSize').innerHTML = `${bytes} bytes`;  
11. }
```

```
1. const handleFiles = (files) => {
2.   for (let i = 0; i < files.length; i++) {
3.     const file = files[i];
4.
5.     if (file.type.startsWith('image/')) {
6.       const img = document.createElement('img');
7.       preview.appendChild(img);
8.
9.       const reader = new FileReader();
10.      reader.addEventListener('load', (event) => {
11.        img.src = event.target.result;
12.      });
13.
14.      reader.readAsDataURL(file);
15.    }
16.  }
17.}
```

- `FileReader.readAsArrayBuffer()`
- `FileReader.readAsBinaryString()`
- `FileReader.readAsDataURL()`
- `FileReader.readAsText()`

1. `const objectURL = window.URL.createObjectURL(fileObj);`
2. `window.URL.revokeObjectURL(objectURL);`

<https://codepen.io/priver/pen/xxxXwWJ>



```
1. fetch('http://www.example.com', {  
2.     method: 'POST',  
3.     body: file,  
4. });
```

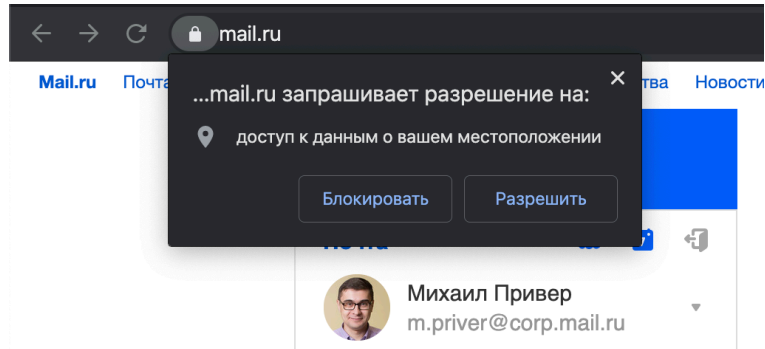
```
1. const data = new FormData();  
2. data.append('file', file);  
3. data.append('user', 'Ivan');
```

```
5. fetch('http://www.example.com', {  
6.     method: 'POST',  
7.     body: data,  
8. });
```

- `navigator.geolocation`
- Доступно только для HTTPS

```
1. if ("geolocation" in navigator) {  
2.     // Геолокация доступна  
3. } else {  
4.     // Геолокация недоступна  
5. }
```

```
1. navigator.geolocation.getCurrentPosition((position) => {  
2.     doSomething(position.coords.latitude, position.coords.longitude);  
3. });
```



```
1. position: Position
2.   timestamp: 1573137881319
3.   coords: Coordinates
4.     accuracy: 721220
5.     altitude: null
6.     altitudeAccuracy: null
7.     heading: null
8.     latitude: 55.823586799999994
9.     longitude: 37.5582644
10.    speed: null
```

```
1. const watchID = navigator.geolocation.watchPosition((position) => {  
2.     do_something(position.coords.latitude, position.coords.longitude);  
3. });  
4.  
5. navigator.geolocation.clearWatch(watchID);
```

```
1. const geoSuccess = (position) => {  
2.     doSomething(position.coords.latitude, position.coords.longitude);  
3. };  
4.  
5. const geoError = (error) => {  
6.     console.log(error.message);  
7. };  
8.  
9. var geoOptions = {  
10.     enableHighAccuracy: true,  
11.     maximumAge: 30000,  
12.     timeout: 27000,  
13. };  
14.  
15. navigator.geolocation.getCurrentPosition(geoSuccess, geoError, geoOptions);
```

```
1. <div id="columns">
2.     <div class="column" draggable="true"><header>A</header></div>
3.     <div class="column" draggable="true"><header>B</header></div>
4.     <div class="column" draggable="true"><header>C</header></div>
5. </div>
```

<https://www.html5rocks.com/ru/tutorials/dnd/basics/>

- **dragstart** — Срабатывает когда элемент начал перемещаться.
- **drag** — запускается при перемещении элемента или выделенного текста.
- **dragenter** — срабатывает, когда перемещаемый элемент попадает на элемент-назначение.
- **dragleave** — запускается в момент перетаскивания, когда курсор мыши выходит за пределы элемента.
- **dragover** — срабатывает каждые несколько сотен миллисекунд, когда перемещаемый элемент оказывается над зоной, принимающей перетаскиваемые элементы.
- **drop** — Событие drop вызывается для элемента, над которым произошло "сбрасывание" перемещаемого элемента.
- **dragend** — операция перетаскивания завершена (отпустили кнопку мыши; нажали Esc).



```
1. const handleDragStart = (event) => {  
2.     event.target.style.opacity = '0.4';  
3. };  
4.  
5. const cols = Array.from(document.querySelectorAll('#columns .column'));  
6. cols.forEach((col) => {  
7.     col.addEventListener('dragstart', handleDragStart, false);  
8. });
```

```
1. const handleDragStart = (event) => {
2.     event.target.style.opacity = '0.4';
3. };
4.
5. const handleDragOver = (event) => {
6.     event.preventDefault();
7.     event.dataTransfer.dropEffect = 'move';
8. };
9.
10. const handleDragEnter = (event) => {
11.     event.target.classList.add('over');
12. };
13.
14. const handleDragLeave = (event) => {
15.     event.target.classList.remove('over');
16. };
17.
18. const cols = Array.from(document.querySelectorAll('#columns .column'));
19. cols.forEach((col) => {
20.     col.addEventListener('dragstart', handleDragStart, false);
21.     col.addEventListener('dragenter', handleDragEnter, false);
22.     col.addEventListener('dragover', handleDragOver, false);
23.     col.addEventListener('dragleave', handleDragLeave, false);
24. });
```

```
1. ...
2.
3. const handleDrop = (event) => {
4.     event.stopPropagation();
5.     event.preventDefault();
6. };
7.
8. const handleDragEnd = (event) => {
9.     event.target.style.opacity = 1;
10.    Array.from(document.querySelectorAll('#columns .column'));
11.    cols.forEach((col) => {
12.        col.classList.remove('over');
13.    });
14. };
15.
16. const cols = Array.from(document.querySelectorAll('#columns .column'));
17. cols.forEach((col) => {
18.     ...
19.     col.addEventListener('drop', handleDrop, false);
20.     col.addEventListener('dragend', handleDragEnd, false);
21. });
```

- **dataTransfer.effectAllowed** — ограничивает "тип перетаскивания", которое пользователь может выполнять с элементом. Это свойство используется в модели обработки перетаскивания для инициализации объекта `dropEffect` во время событий `dragenter` и `dragover`. Это свойства может принимать следующие значения: `none`, `copy`, `copyLink`, `copyMove`, `link`, `linkMove`, `move`, `all` и `uninitialized`.
- **dataTransfer.dropEffect** — управляет реакцией, которую пользователь получает во время событий `dragenter` и `dragover`. Когда перетаскиваемый объект наводится на целевой элемент, указатель браузера принимает вид, соответствующий типу предполагаемой операции (например, копирование, перенос и т. д.). Свойство может принимать следующие значения: `none`, `copy`, `link`, `move`.
- **dataTransfer.setDragImage(img element, x, y)** — вместо использования "фантомного изображения", которое браузер создает по умолчанию, можно задать значок перетаскивания.

```
1. const dropbox = document.getElementById('dropbox');
2.
3. const preventAndStop = (event) => {
4.     event.stopPropagation();
5.     event.preventDefault();
6. };
7.
8. const drop = (event) => {
9.     preventAndStop(event);
10.    const files = event.dataTransfer.files;
11.    handleFiles(files);
12. }
13.
14. dropbox.addEventListener('dragenter', preventAndStop, false);
15. dropbox.addEventListener('dragover', preventAndStop, false);
16. dropbox.addEventListener('drop', drop, false);
```

```
1. <audio
2.     controls
3.     src="/media/examples/t-rex-roar.mp3">
4. </audio>
5.
6. <audio id="music" controls>
7.     <source src="myAudio.mp3" type="audio/mpeg">
8.     <source src="myAudio.ogg" type="audio/ogg">
9.     <p>Ваш браузер не поддерживает воспроизведение.</p>
10. </audio>
```

```
1. const audio = document.getElementById('music');
2.
3. audio.currentTime = 10;
4. audio.volume = (Math.exp(0.5) - 1) / (Math.E - 1);
5.
6. audio.play();
7. audio.pause();
8.
9. audio.addEventListener('canplay', canPlayHandler);
10. audio.addEventListener('canplaythrough', canPlayThroughHandler);
11. audio.addEventListener('ended', endedHandler);
12. audio.addEventListener('error', errorHandler);
13. audio.addEventListener('timeupdate', timeUpdateHandler);
```

## Audio Context





```
1. const audioContext = new AudioContext();
2. const audioElement = document.querySelector('audio');
3. const track = audioContext.createMediaElementSource(audioElement);
4. track.connect(audioContext.destination);
5.
6. let isPlaying = false;
7.
8. const playButton = document.querySelector('button');
9. playButton.addEventListener('click', function() {
10.     if (audioContext.state === 'suspended') {
11.         audioContext.resume();
12.     }
13.
14.     // play or pause track depending on state
15.     if (!isPlaying) {
16.         audioElement.play();
17.         isPlaying = true;
18.     } else {
19.         audioElement.pause();
20.         isPlaying = false;
21.     }
22.
23. }, false);
24.
```

```
1. const gainNode = audioContext.createGain();  
2. track.connect(gainNode).connect(audioContext.destination);  
3. gainNode.gain.value = 0.5;  
4.  
5. const panner = new StereoPannerNode(audioContext, { pan: 0 });  
6. track.connect(panner).connect(audioContext.destination);  
7. panner.pan.value = -1;
```

<https://codepen.io/nfj525/pen/rVBaab>

```
1. async function getMedia() {  
2.   let stream = null;  
  
4.   try {  
5.     const constrains = { audio: true, video: true };  
6.     stream = await navigator.mediaDevices.getUserMedia(constraints);  
7.     // можно использовать stream  
8.   } catch(err) {  
9.     // обработка ошибки  
10.  }  
11. }
```

<https://codepen.io/snapppy/pen/zdXvvP>

```
1. const mediaRecorder = new MediaRecorder(stream);
2.
3. mediaRecorder.start();
4. mediaRecorder.stop();
5.
6. const chunks = [];
7. mediaRecorder.addEventListener('stop', (event) => {
8.     const audio = document.createElement('audio');
9.     const blob = new Blob(chunks, { type: mediaRecorder.mimeType });
10.    chunks = [];
11.    const audioURL = URL.createObjectURL(blob);
12.    audio.src = audioURL;
13. });

15. mediaRecorder.addEventListener('dataavailable', (event) => {
16.    chunks.push(event.data);
17. });
18.
```

1. `window.navigator.vibrate(200);`
2. `window.navigator.vibrate([200, 100, 200]);`

<https://googlechrome.github.io/samples/paymentrequest/credit-cards/>

- Отправка геопозиции
  - Отправка картинок
  - Отправка аудиосообщений
- 
- Срок сдачи: ~ 26 ноября

# Спасибо за внимание!

Михаил Привер

[m.priver@corp.mail.ru](mailto:m.priver@corp.mail.ru)

telegram: @priver