

Лекция 8. События. Браузерные API. Интерфейсы “ввода”

Мартин Комитски



План на сегодня

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

Минутка бюрократии

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях



Event Loop

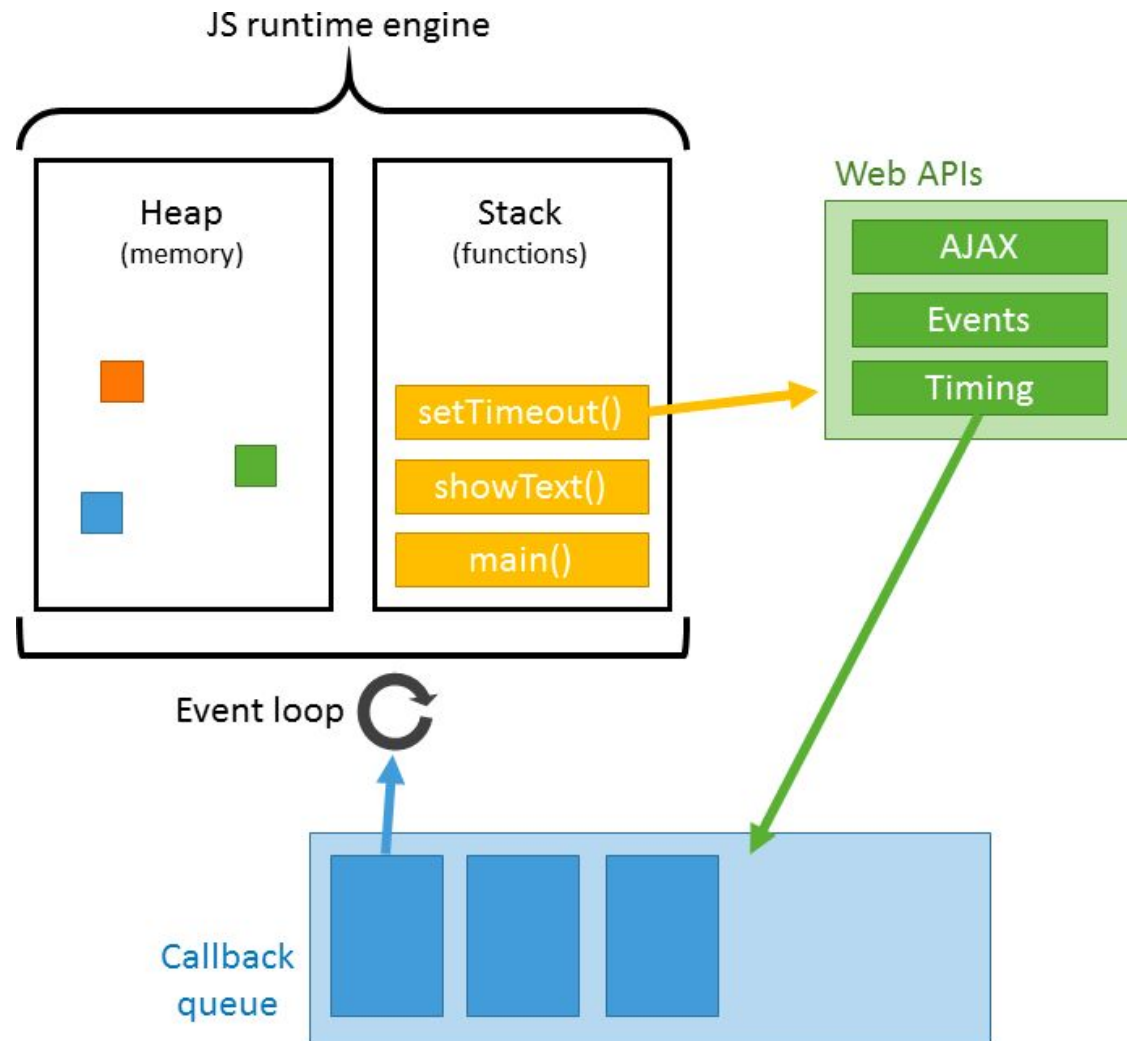
Event Loop. Call Stack

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

Event Loop. Call Stack

- Запуск функции создает контекст выполнения
- Внешний вызов регистрируется в очереди
- Обработчик создает начальный контекст
- Если стек вызова свободен – запускается следующий обработчик

Концепция событийного цикла



Event Loop

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


Event Loop. Очереди событий

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

Event Loop. Очереди событий

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

Event Loop. Микрзадачі

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

Event Loop. Пример

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

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

<http://latentflip.com/loupe/?code=JC5vbignYnV0dG9uJywgJ2NsaWNrJywgZnVuY3Rpb24gb25DbGljaygpIHsKICAgIHNIldFRpbWVvdXQoZnVuY3Rpb24gdGltZXIloKSB7CiAgICAglCAgY29uc29sZS5sb2coJ1lvdSBjbGlja2VklHRoZSBidXR0b24hJyk7ICAglAogICAglCAgY29uc29sZS5sb2colkhplISlpOwoKc2V0VGltZW91dChmdW5jdGlviB0aW1lb3V0KCkgewogICAglCAgY29uc29sZS5sb2colkNsaWNrIHRoZSBidXR0b24hlik7Cn0sIDUwMDApOwoKY29uc29sZS5sb2colldlbGNvbWUgdG8gbG91cGUulik7!!!PGJ1dHRvbj5DbGljayBtZSE8L2J1dHRvbj4%3D>

Event Loop?

Вопросы?



События DOM

События DOM

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-переход завершен
- **animationend** – когда CSS-анимация завершена

События DOM. Назначение обработчика

1. `<input value="Нажми меня" onclick="alert('Клик!')" type="button">`
2. `<input id="elem" type="button" value="Нажми меня!">`
- 3.
4. `<script>`
5. `document.getElementById('elem').onclick = () => {`
6. `alert('Спасибо');`
7. `};`
8. `</script>`

События DOM. Назначение обработчика

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

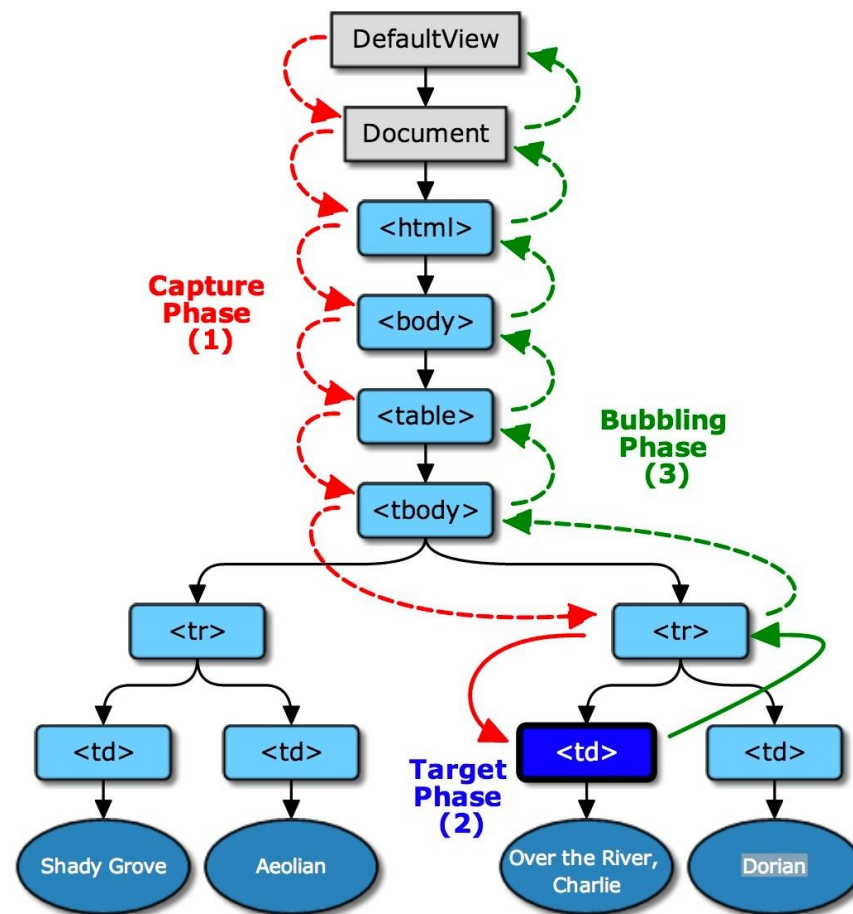
События DOM. Удаление обработчика

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

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

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

События DOM. Схема



События DOM

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

<https://learn.javascript.ru/bubbling-and-capturing>

События DOM. Объект события

Использование в обработчиках

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

События DOM. Создание своих событий

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 (event) { ... });`

3. `elem.dispatchEvent(event);`

File API

File API. Использование

Возможности:

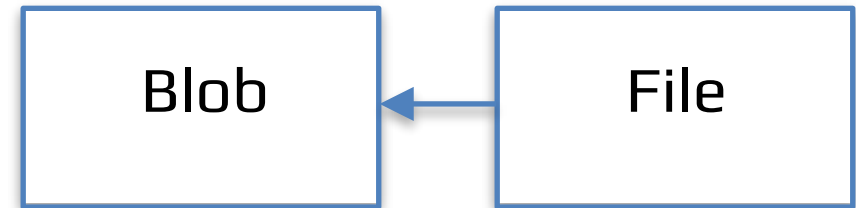
- Получение мета-информации
- Чтение содержимого
- URL-схема

Совместимые интерфейсы:

- `HTMLInputElement.files`
- `XMLHttpRequest.response`, `FormData.append`
- `dataTransfer.files`

File API. Объекты

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



File API. Использование

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

File API. Пример: размер файлов

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

File API. FileReader

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

File API. FileReader: методы

1. `FileReader.readAsArrayBuffer()`
2. `FileReader.readAsBinaryString()`
3. `FileReader.readAsDataURL()`
4. `FileReader.readAsText()`

File API. Object URLs

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

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

File API. Загрузка файлов на сервер

```
1. fetch('http://www.example.com', {  
2.   method: 'POST',  
3.   body: file,  
4. });  
5.  
6. const data = new FormData();  
7. data.append('file', file);  
8. data.append('user', 'Ivan');  
9.  
10. fetch('http://www.example.com', {  
11.   method: 'POST',  
12.   body: data,  
13. });  
14.
```



Перерыв! (10 минут)

Препо (с)

Geolocation API

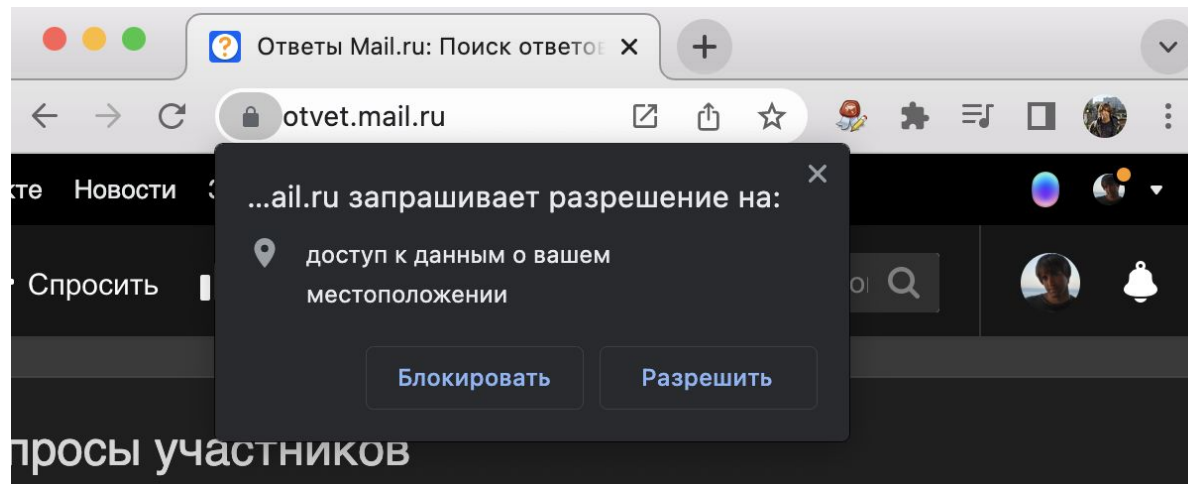
Geolocation API

- Запрос разрешения
- Асинхронная операция
- Разная степень точности (используется не только GPS)
- Доступно только для HTTPS

```
1. navigator.geolocation
2. if ('geolocation' in navigator) {
3.   // Геолокация доступна
4. } else {
5.   // Геолокация недоступна
6. }
```

Geolocation API. `getCurrentPosition`

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



Geolocation API. Объекты Position и Coordinates

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

Geolocation API. watchPosition

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

Geolocation API. Дополнительные параметры

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

Drag and Drop

Drag and Drop

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://web.dev/drag-and-drop/>

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

Drag and Drop. События

- **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).

Drag and Drop. dragstart

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

Drag and Drop. dragenter, dragover, dragleave

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

Drag and Drop. drop, dragend

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

Drag and Drop. `dataTransfer`

- **`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.

Drag and Drop. Файлы

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

Notification API

Notification API

```
1. const permission = await Notification.requestPermission();  
2.  
3. const greeting = new Notification('Hi, How are you?');  
4.  
5. const greeting = new Notification('Hi, How are you?', {  
6.   body: 'Have a good day',  
7.   icon: './img/gooodday.png'  
8. });  
9.  
10. greeting.close();  
11.
```

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

Web Audio API

Web Audio API. Элемент <audio>

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

Web Audio API. HTMLMediaElement

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

https://developer.mozilla.org/en-US/docs/Web/API/HTMLMediaElement/canplay_event

Web Audio API

Audio Context



Web Audio API. Воспроизведение

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

Web Audio API. Модификация

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>

Media Devices

Media Devices

```
1. async function getMedia() {  
2.   let stream = null;  
3.  
4.   try {  
5.     const constraints = { 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>

MediaRecorder

```
1.  const mediaRecorder = new MediaRecorder(stream);
2.  mediaRecorder.start();
3.  mediaRecorder.stop();
4.  const chunks = [];
5.
6.  mediaRecorder.addEventListener('stop', (event) => {
7.    const audio = document.createElement('audio');
8.    const blob = new Blob(chunks, { type: mediaRecorder.mimeType });
9.    chunks = [];
10.   const audioURL = URL.createObjectURL(blob);
11.   audio.src = audioURL;
12. });
13.
14. mediaRecorder.addEventListener('dataavailable', (event) => {
15.   chunks.push(event.data);
16. });
```

Другие API

Vibration API

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

Payment API

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

Другие API

<https://whatwebcando.today/>

Домашнее задание №8

1. Отправка геопозиции
2. Отправка картинок
3. Отправка аудиосообщений
4. Получение уведомлений, звукового сопровождения
5. Вибрацию и др. по желанию

Расширенное описание задания, подсказки, а также презентации с лекций всегда есть в репозитории.

Срок сдачи

1 декабря

Спасибо за внимание!



Пока!

Присоединяйтесь к сообществам про образование в VK

- [VK Джуниор](#)
- [VK Образование](#)

