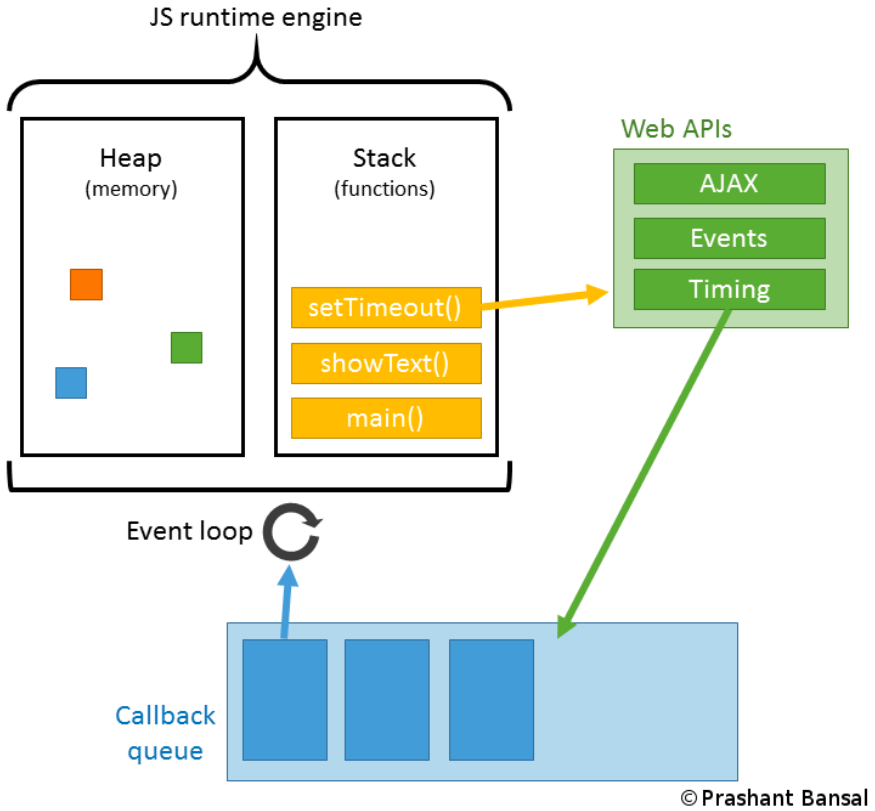
Интерфейсы ввода

Борис Ребров

Минутка бюрократии

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

Event loop



Event loop

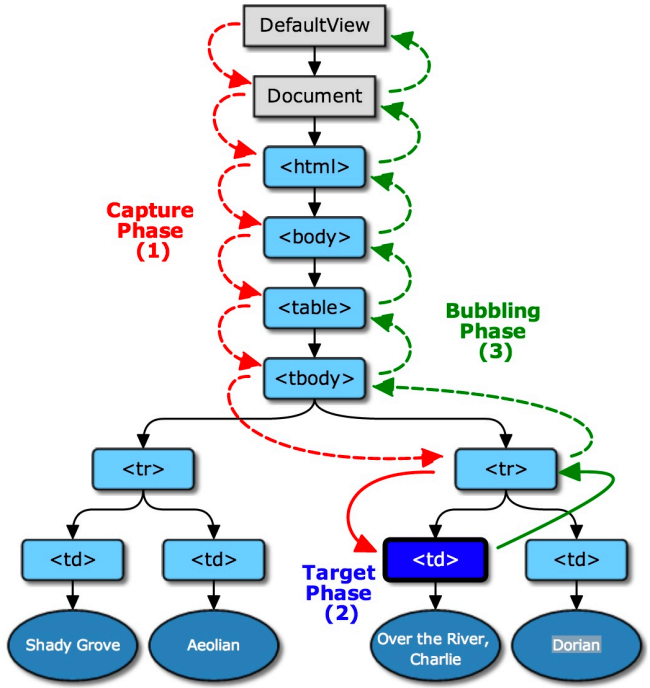
- Запуск функции создает контекст выполнения
- Внешний вызов регистрируется в очереди
- Обработчик создает начальный контекст
- Если стэк свободен – запускается следующий обработчик

```
while(queue.waitForMessage()){
  queue.processNextMessage();
}
```

```
setTimeout(function () {
    console.log(true);
}, 0);

console.log(false);
```

События DOM


DefaultView
Document
<html>
<body>
<table>
<tbody>
<tr>
<tr>
<td>
<td>
<td>
<td>
Shady Grove
Aeolian
Over the River, Charlie
Dorian
Capture Phase (1)
Bubbling Phase (3)
Target Phase (2)

События DOM

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

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
  - можно назначать инлайн

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

```
1. function handler2 () {
2.  console.log('handler 2');
3. }
4. $0.addEventListener('click', function handler1 () {
5.  this.removeEventListener('click', handler2);
6.  console.log('handler 1');
7. });
8. $0.addEventListener('click', handler2);
```

События DOM: порождение

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

- Принимает объект события

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

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

События DOM: объект события

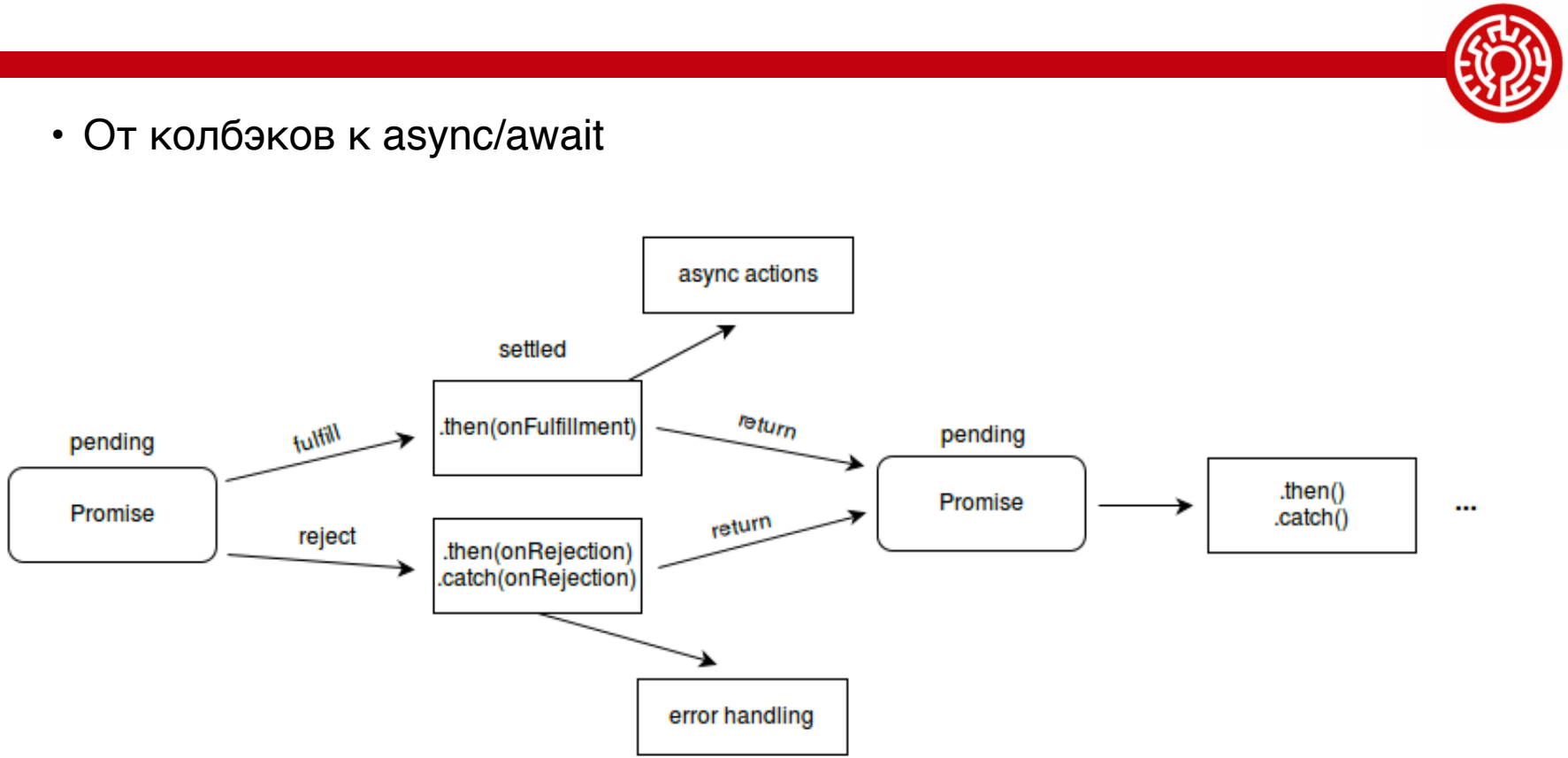
- Тип
- Фаза обработки
- Элемент, для которого назначен обработчик
- Элемент, на котором захвачено событие
- Timestamp, информацию о всплытии и обработке по умолчанию

```
1.  event = new Event(typeArg[, eventInit]);
```

- Легаси

```
1. var event = document.createEvent('MouseEvent');
2. event.initEvent('click', true, true);
```

Promise



- От колбэков к async/await

Promise

- Создание обещаний

```
1. new Promise(function(resolve, reject) { ... });
3. Promise.all(iterable);
5. Promise.race(iterable);
7. Promise.reject(reason);
9. Promise.resolve(reason);
```

Promise

- Обработка обещаний

```
1.  Promise.prototype.catch(onRejected);

3.  Promise.prototype.then(onFulfilled, onRejected)
```

```
1.  var rejectedPromise = Promise.reject('error');
2.  var fullfiledPromise = rejectedPromise.catch(reason => 'success');

3.  fullfiledPromise.then(console.log); //'success'
```

Promise

- Применение

```
1.  function loadImage (src) {
2.      return new Promise((resolve, reject) => {
3.          imageElem = new Image();
4.          imageElem.onload = event => resolve(imageElem);
5.          imageElem.onerror = reject;
6.          imageElem.src = src;
7.      });
8.  }
```

```
1.  function loadImages (imagesList) {
2.      return Promise.all(imagesList.map(loadImage));
3.  }

loadImages([
    'https://picsum.photos/200/300/?random',
    'https://picsum.photos/g/200/300'
]).then(console.log)
```

Geolocation

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

```
{
    accuracy: 20
    altitude: null
    altitudeAccuracy: null
    heading: null
    latitude: 55.797058
    longitude: 37.5378234
    speed: null
},
timestamp: 1539597250899
```

Geolocation

- Получение позиции

```
1. navigator.geolocation.getCurrentPosition(console.log, console.err, {
2.   enableHighAccuracy: true,
3.   timeout: 5000,
4.   maximumAge: 0
5. });
```

- Слежение за позицией

```
1. var watchId = navigator.geolocation.watchPosition(console.log, console.err);
2. navigator.geolocation.clearWatch(watchId);
```

FileAPI

- Возможности:
  - Получение мета-информации
  - Чтение содержимого
  - URL-схема
- Совместимые интерфейсы
  - HTMLInputElement.files
  - XMLHttpRequest.response, FormData.append
  - dataTransfer.files

FileAPI

- FileList

```
1. inputElem.files.length
```

- Blob

```
1. var blob = new Blob([JSON.stringify({key: ‘value'})],{type : ‘application/json'});
```

- File

```
1. File.prototype.__proto__ // Blob
```

- FileReader

```
1. var reader  = new FileReader();
2. reader.addEventListener("load", …);
3. reader.readAsDataURL(file);
```

FileAPI

- Blob/File

```
1. Blob.size
2. Blob.type
3. Blob.slice([start[, end[, contentType]]])
```

```
1. File.lastModified
2. File.name
```

```
1. var myFile = new File(
2.     [JSON.stringify({key: ‘value’})],
3.     {type : ‘application/json’}
4. );
5. window.open(URL.createObjectURL(myFile));
```

FileAPI

- URL
  - работает почти как обычный
  - умеет только GET
  - может быть отозван

```
1. var myFile = document.querySelector(‘input[type=file]’).files[0];
2. var image = new Image;
3. var url = URL.createObjectURL(myFile);
4. image.onload = () => URL.revokeObjectURL(url);
5. image.src = url;
```

FileAPI

- FileReader

```
1.  var myFile = document.querySelector(‘input[type=file]’).files[0];
2.  var image = new Image;
3.  var reader  = new FileReader();
4.  reader.addEventListener("load", () => {
      image.src = reader.result;
}
reader.readAsDataUrl(myFile);
```

Домашнее задание № 3

- Ознакомиться с документацией по ссылкам
- Добавить в форму веб-компоненты для запроса геопозиции и выбора файла, при выборе файла отображать его preview.
- Реализовать возможность выбора файла с помощью Drag&Drop

Срок сдачи

- ~ 22 октября

Спасибо за внимание!