



{ Асинхронный JavaScript }

Дмитрий Нефедов

dnefedov@sportmaster.ru

```
.className {  
  transition: color 10s linear;  
}
```

```
function animate(color) {  
  element.style.color = color;  
}
```

```
animate("blue");
```

```
setTimeout(() => animate("red"), 10);
```

.className



{ JavaScript }
однопоточный
не блокирующий
асинхронный



{ Концепция жизненного цикла }



{ Stack }

структура, описывающая порядок выполнения кода



{ Heap }

это ссылка на определённую неструктурированную область
памяти

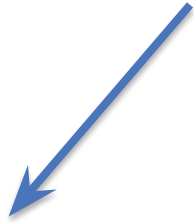
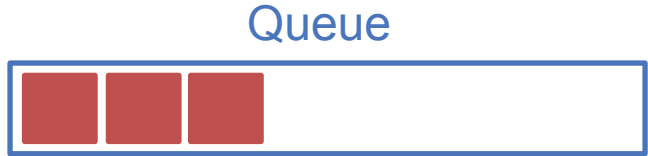
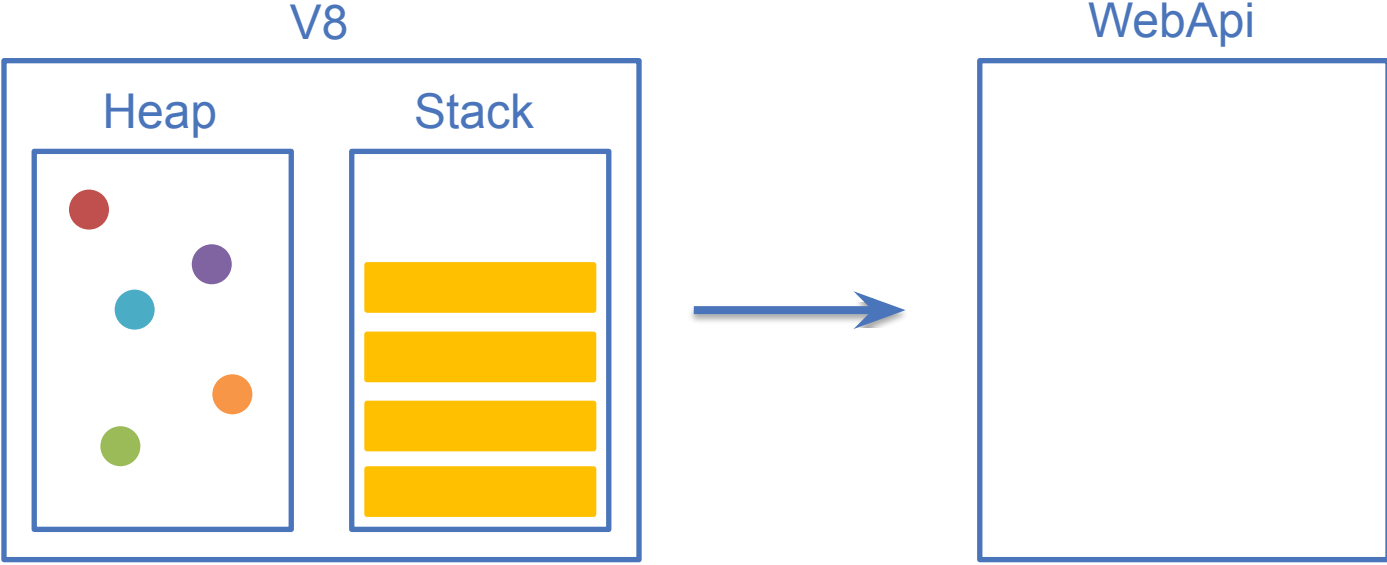


{ Event Queue }

список событий, подлежащих обработке

Каждое событие ассоциируется с некоторой функцией. Когда на стеке освобождается достаточно места, событие извлекается из очереди и обрабатывается






```
function fizz() {  
  throw 'Error';  
}  
  
function bazz() {  
  fizz();  
}  
  
function fizzbazz() {  
  fizz();  
  bazz();  
}  
  
fizzbazz();
```

Stack



```
▼ Uncaught Error  
fizz      @ VM171:2  
fizzbazz  @ VM171:8  
(anonymous) @ VM171:11
```



{ Переполнение стека }



```
function bazz() {  
  bazz();  
}
```

```
bazz();
```

```
▶ Uncaught RangeError: Maximum call stack size exceeded  
  at bazz (<anonymous>:1:14)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)  
  at bazz (<anonymous>:2:3)
```



{ Очередь событий }



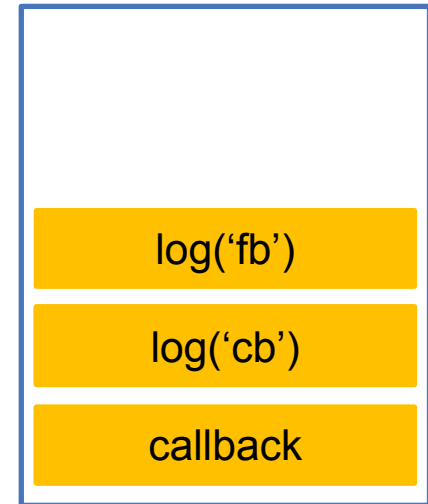
```
function fizzbuzz() {  
  console.log('fb');  
}
```

```
function callback() {  
  console.log('cb');  
}
```

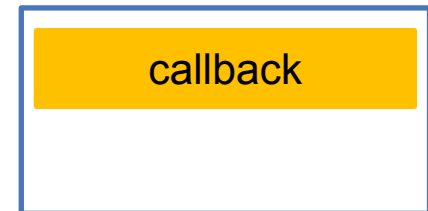
```
setTimeout(callback, 5000);
```

```
fizzbuzz();
```

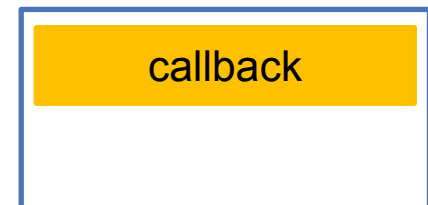
Stack



WebApi



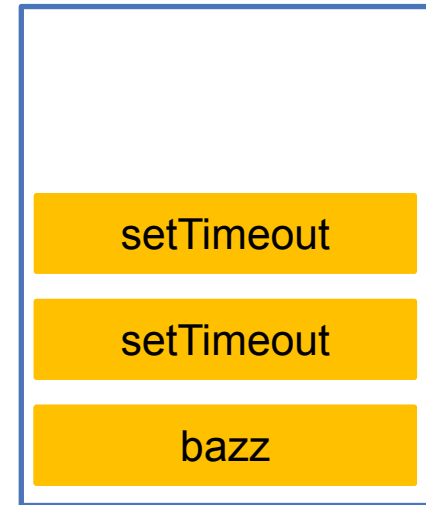
Queue



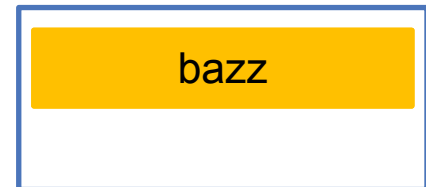
```
function bazz() {  
  setTimeout(bazz, 0);  
}
```

```
bazz();
```

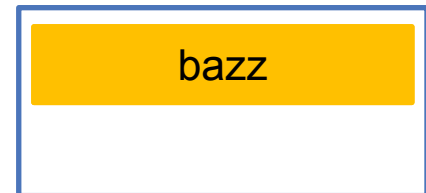
Stack



WebApi



Queue

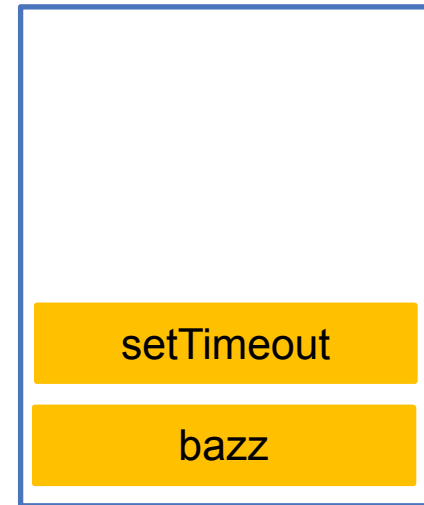


```
function bazz() {  
  delay(5000);  
}
```

```
setTimeout(bazz, 0);
```

```
setTimeout(bazz, 0);
```

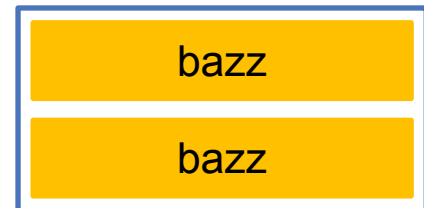
Stack

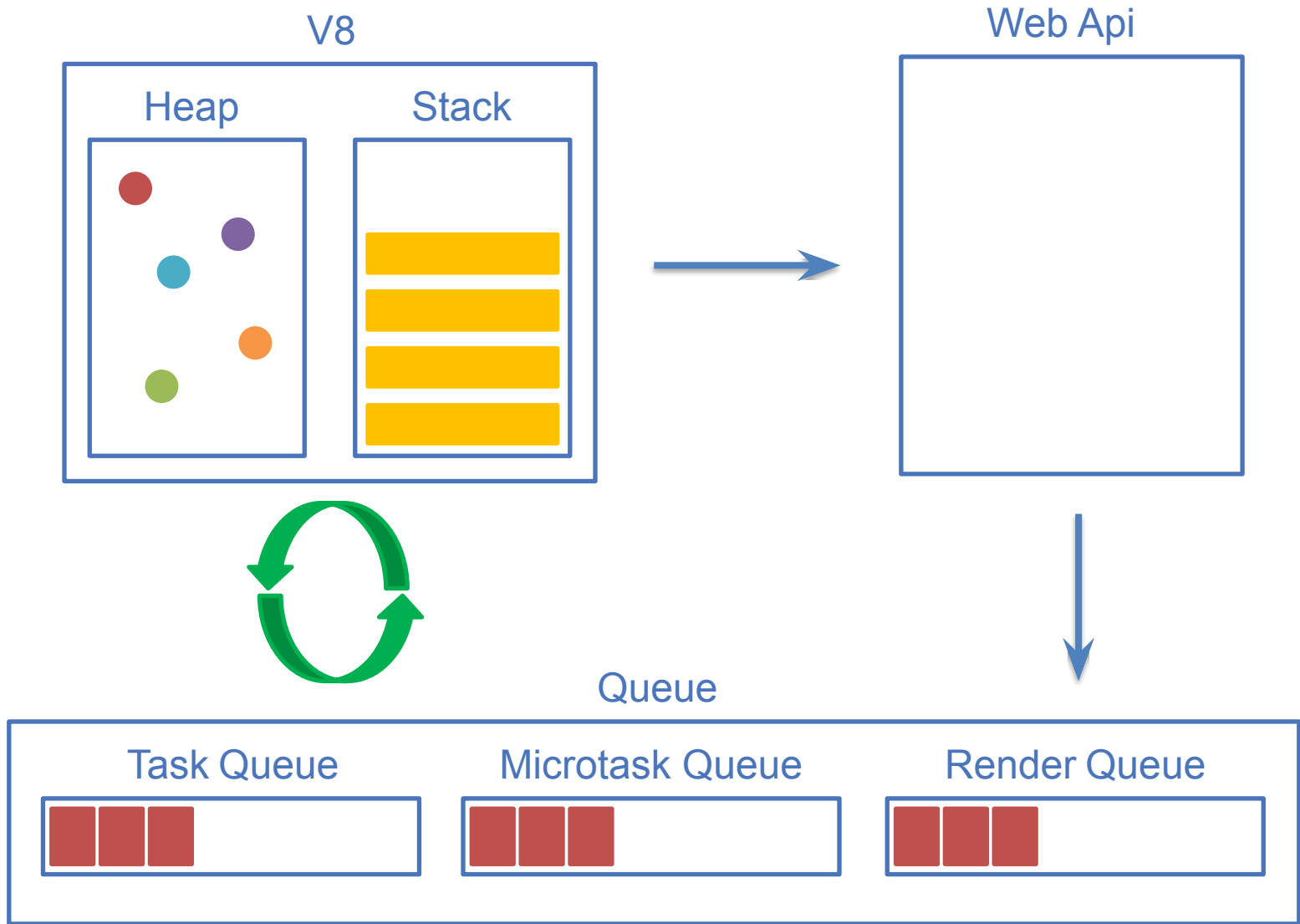


WebApi



Queue

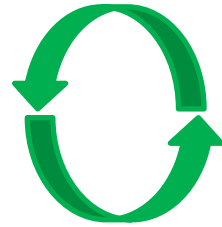
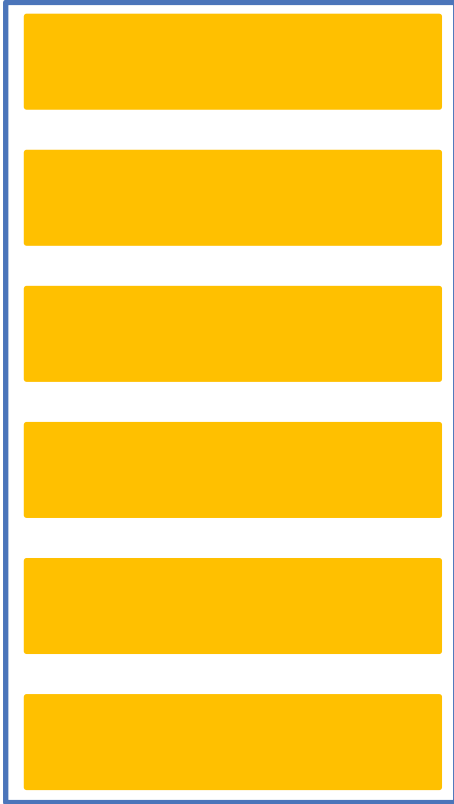




{ Task Queue }



Task Queue



Stack



{ Microtask Queue }



```
button.addEventListener('click', () => {  
  Promise.resolve().then(() => console.log('Promise 1'));  
  console.log('Click 1');  
});
```

```
button.addEventListener('click', () => {  
  Promise.resolve().then(() => console.log('Promise 2'));  
  console.log('Click 2');  
});
```

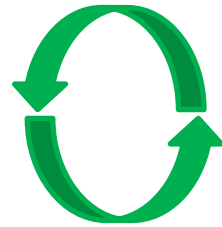
```
Click 1  
Promise 1  
Click 2  
Promise 2
```

```
button.click();
```

```
Click 1  
Click 2  
Promise 1  
Promise 2
```



Microtask Queue

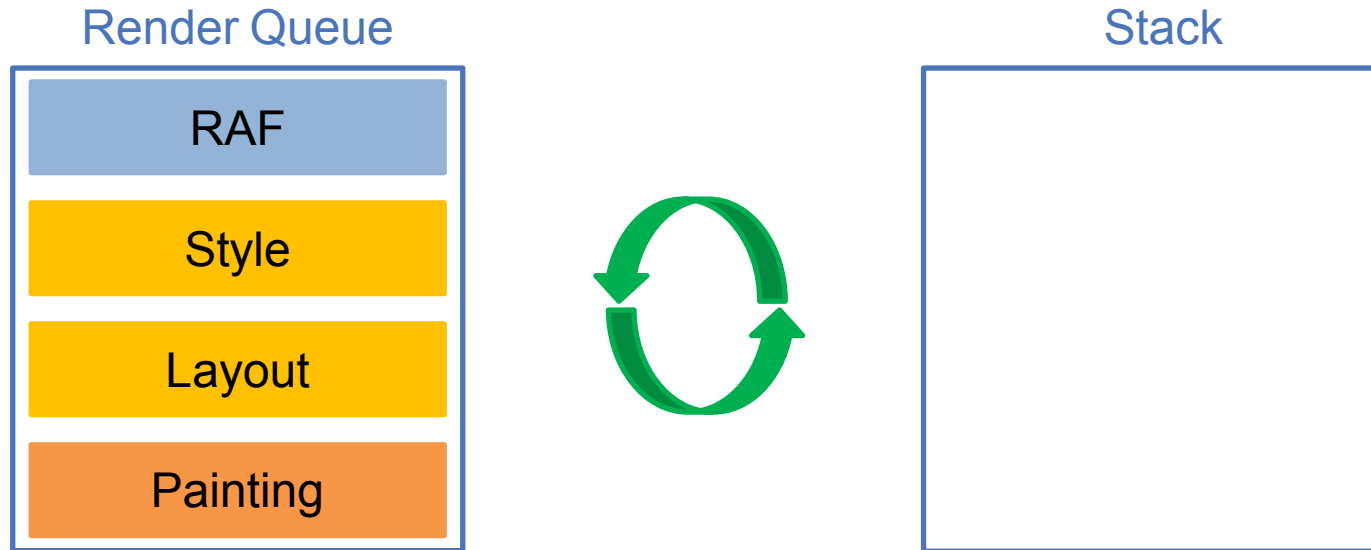


Stack



{ Render Queue }





```
button.addEventListener('click', () => {  
  element.style.color = 'red';  
  element.style.color = 'black';  
  element.style.color = 'red';  
  element.style.color = 'black';  
  element.style.color = 'red';  
  element.style.color = 'black';  
});
```



```
button.addEventListener('click', () => {  
  element.style.color = 'red';  
  element.style.transition = 'color 1s linear';  
  element.style.color = 'black';  
});
```

```
button.addEventListener('click', () => {  
  element.style.color = 'red';  
  element.style.transition = 'color 1s linear';  
  requestAnimationFrame(() => {  
    element.style.color = 'black';  
  });  
});
```




```
button.addEventListener('click', () => {  
  element.style.color = 'red';  
  element.style.transition = 'color 1s linear';  
  requestAnimationFrame(() => {  
    requestAnimationFrame(() => {  
      element.style.color = 'black';  
    });  
  });  
});
```

```
button.addEventListener('click', () => {  
  element.style.color = 'red';  
  element.style.transition = 'color 1s linear';  
  getComputedStyle(element).color;  
  element.style.color = 'black';  
});
```



Спасибо за внимание!

