

# Библиотека рекламных форматов

Это единая JS библиотека, которая используется для показа следующих форматов рекламы с использованием адсервера AdFox:

- Adhesion
- Billboard
- Flowbutton
- Fullscreen
- Infinite Inread
- Inread
- Interscroller
- Wrapper
- Sticker

Библиотека позволяет использовать кастомные шаблоны баннеров и HeaderBidding с подключением сторонних вендоров

## Руководство по подключению библиотеки рекламных форматов:

Подключение библиотеки рекламных форматов производится путем добавления в <head> сайта тега <script> со ссылкой на желаемую версию библиотеки.

Различные версии библиотеки находятся в директориях, соответствующих номеру сборки. Например, <https://s3.wi-fi.ru/mtt/banners/libs/1.4.3/all.js>

Возможен как синхронный, так и асинхронный вариант подключения:

### Синхронное подключение:

В тело страницы необходимо включить

```
<script src="https://s3.wi-fi.ru/mtt/banners/libs/latest/all.js"></script>  
<script>// код вызова </script>
```

### Асинхронное подключение:

```
<script>  
new Promise((resolve, reject) => {  
  const lib = document.createElement("script");  
  lib.type = "application/javascript";  
  lib.src = "https://s3.wi-fi.ru/mtt/banners/libs/latest/all.js";  
  lib.async = true;  
  lib.addEventListener("load", resolve);  
  lib.addEventListener("error", reject);  
  document.head.appendChild(lib);  
})  
  .then(() => {  
    // код вызова  
  })  
  .catch(() => {  
    // обработка ошибки  
  });  
</script>
```

**Чтобы добавить на сайт релизный скрипт баннеров, нужно выполнить следующий скрипт:**

```
document.head.appendChild((function() {  
  const s = document.createElement('script');  
  s.src = 'http://static.wi-fi.ru/mtt/banners/libs/release/all.js';  
  s.id='MTBANNERZ';  
  return s;  
})();
```

**Удаление рекламных плейсментов со страницы:**

Удаление всех баннеров

```
document.querySelectorAll('[class^=mtt-]').forEach(e => e.remove());
```

Общий принцип удаления элементов определенного типа: передача в селектор class вида  
[class^=тип-баннера]

Например:

```
document.querySelectorAll('[class^=mtt-inread]').forEach(e => e.remove());  
document.querySelectorAll('[class^=mtt-billboard]').forEach(e => e.remove());  
document.querySelectorAll('[class^=mtt-fullscreen]').forEach(e => e.remove());
```

# Формат Adhesion

## Схема:

```
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
type AdUnit = { code?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}
```

```
type Position = {  
  top: "top",  
  bottom: "bottom"  
}
```

```
enum OwnEvents {  
  SHOW_BANNER = 'SHOW_BANNER',  
  CLOSE_BANNER = 'CLOSE_BANNER',  
  SLIDE_IN = 'SLIDE_IN',  
  SLIDE_OUT = 'SLIDE_OUT',  
  LOADED = 'LOADED',  
}
```

```
enum CreationEvents {  
  LOAD = 'ADFOX.LOAD',  
  RENDER = 'ADFOX.RENDER',  
  STUB = 'ADFOX.STUB',  
  ERROR = 'ADFOX.ERROR',  
}
```

```
enum Core.Events {  
  DESTROY = 'DESTROY',  
}
```

```
type Events = OwnEvents | CreationEvents | Core.Events;
```

```
window.MTT.adhesion({  
  adfox: {  
    ownerId: number,  
    params?: Campaign,  
    paramsAdaptive?: DeviceDepends<Campaign>, // Настройки кампании в зависимости от устройства  
  },  
  adfoxHeaderBidding?: {  
    units: AdUnit[],  
  },  
  banner?: {  
    backgroundColor?: string; // Добавляет подложку во всю ширину страницы с заданным цветом в  
    формате HEX  
    position?: Position; // Определяет позицию для фикса баннера: прокидываем строку "top" || "bottom"  
    minimizable?: boolean; // Заменяет крестик на кнопку свернуть/развернуть и добавляет анимацию  
    появления  
    maxReloads?: number; // Максимальное количество перезагрузок баннера  
  }  
});
```

```
reloadDelay?: number; // Задержка до перезагрузки баннера
skipButtonTimer?: number; // Таймер отсчёта до появления крестика
closeButtonPosition?:
closeButtonPosition; // положение крестика закрытия баннера
```

```
events?: {
  Events: function;
},
});
```

## Структуры верхнего уровня:

### **adfox**

Настройки рекламной компании (общее для всех баннеров)

### **adfoxHeaderBidding?**

Массив ставок передающихся в `window.Ya.headerBidding.pushAdUnits(units);`  
<https://yandex.ru/support/adfox-sites/monetization/header-bidding.html>

### **banner?**

Настройки баннера `adhesion`

### **events?**

Подписки на события `adhesion`a`

## Настройки баннера:

### **banner.position?**

Настройка позволяет выбрать у верхнего или нижнего края экрана должен зафиксироваться баннер.

Обособленный параметр который работает независимо от других настроек баннера  
По умолчанию: "bottom" (нижний край)

### **banner.minimizable?**

Определяет внешний вид и функционал кнопки.

Крест закрывающий баннер, или кнопка. сворачивающая/разворачивающая его.

Параметр напрямую связан с `banner.position!`

Для корректной работы анимаций при использовании кнопки сворачивания - обязательно нужно задать позицию баннера "top" || "bottom"

По умолчанию: false (крест)

### **banner.backgroundColor?**

Настройка позволяет выбрать цвет фона для баннера.

При включении контейнер `adhesion` занимает 100% ширины вьюпорта и баннер находится в центре подложки с заданным цветом.

По умолчанию: Фон не задан и контейнер растянут только на ширину контента

### **banner.reloadDelay?**

Настройка позволяет включить перезагрузку баннера.

При закрытии/сворачивании через заданный таймаут баннер будет вызван заново.

По умолчанию: 60 секунд

### **banner.maxReloads?**

Настройка позволяет включить перезагрузку баннера.

Определяет количество перезагрузок которые может сделать баннер.

По умолчанию: 2 тика

### **banner.skipButtonTimer?**

Настройка позволяет включить отображение обратного отсчёта до появления крестика.

### **banner.closeButtonPosition?**

Настройка определяющая положение крестика закрытия баннера. Принимает параметры

- left | right | left-top | right-top

По умолчанию: closeButtonPosition: "left"

### **Подписки на события баннера:**

#### **events?**

Настройка позволяет подписаться на любое событие.

Подписка происходит в формате Key: Events (все события есть в схеме), Value: callback который отработает по наступлению события.

#### **Destroy:**

В версии 1.8.0 добавлена возможность дестроя баннера снаружи.

Это необходимо для корректной работы в рамках спа-приложений.

```
const adhesion = window.MTT.adhesion({...}) // adhesion вернет публик-контроллер который имеет метод  
destroy: ()=> void  
adhesion.destroy() // этот метод вызовет отписку от всех событий/слушателей и удалит баннер
```

# Формат Billboard

Баннер-растяжка. Закрепляется на месте, заданном в настройках, либо добавляется первым элементом на странице.

Баннер так же может ездить за пользователем, закрепляясь в шапке.

Прилипание баннера к шапке прекращается исходя из настроек см. св-во `banner.exit`.

Далее баннер остается на своем месте.

## Схема:

```
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
window.MTT.billboard({  
  adfox: {  
    ownerId: number  
    params?: Campaign  
    paramsAdaptive?: DeviceDepends<Campaign>  
  },  
  banner?: {  
    exit?: { // Механика исчезновения прилипшего баннера  
      type: ENUM('draw_ad', 'page_length') // тип, соответственно (время после отрисовки, по длине  
страницы)  
      value?: number, // Значение для конкретного типа, соответственно (5000 (мс), 30 (%))  
    },  
    insertInto?: HTMLElement, // Элемент, куда будет вставлен билборд на свое статичное место.  
    widthConstraints?: { // Запрет прилипания на устройствах с шириной экрана N  
      min?: number, // px  
      max?: number // px  
    }  
  }  
});
```

## Структуры верхнего уровня:

### **adfox**

настройки рекламной компании (общее для всех баннеров)

### **banner**

настройки баннера билборд (необязательный)

## Настройки баннера:

### **banner.exit?**

Механика исчезновения прилипшего баннера.

Свойство	Тип	Описание
type	ENUM('draw_ad', 'page_length')	Тип, соответственно (время после отрисовки, по длине страницы)
value?	number any	Значение для конкретного типа, соответственно 5000 (мс) или 30 (%)

По дефолту стоит значение draw\_ad на 5000 мс

Если задан тип page\_length, но не задано значение, то значение по умолчанию – 30%

### **banner.insertInto?**

Место вставки статичного контейнера билборда(place), т.е. его место на странице.

По дефолту используется document.body.

Вставка производится так:

```
insertInto.insertAdjacentElement('afterbegin', place)
```

### **banner.widthConstraints?**

Запрет прилипания на устройствах с шириной экрана N

min, max – Параметры, позволяющие отключить прилипание на устройствах, ширина которых не попадает в  $min < width < max$ ;

Должно быть задано хотя бы одно свойство.

### **workMode? : ENUM(string)**

Доступные режимы работы баннера:

стандартный баннер ADFOX, без фиксации и без залипания(standard\_adfox)

баннер с залипанием(sticky)

баннер с функцией фиксации скролла(fix\_scroll)

баннер с фиксацией скролла и залипанием(fix\_sticky)

По умолчанию установлено значение sticky.

### **scrollFixation? : Object**

Фиксация скролла настраивается отдельно для каждого типа устройств(desktop, tablet, mobile) в значении ключа scrollFixation. Кроме того, можно не указывать конкретное устройство, а задать параметры настройки глобально для объекта scrollFixation. В таком случае, настройки будут применены ко всем типам устройств.

Настройка происходит по аналогии с paramsAdaptive.

На данный момент к настраиванию доступны:

время фиксации скролла(timer; по умолчанию - 2 сек; значение необходимо задавать в миллисекундах, например: 2000)

включение/выключение подложки с затемнением(overlay; по умолчанию - true)

эти параметры не являются обязательными.

### **Destroy:**

В версии 1.8.0 добавлена возможность дестроя баннера снаружи.

Это необходимо для корректной работы в рамках спа-приложений.

```
const billboard = window.MTT.billboard({...}) // billboard вернет публик-контроллер который имеет метод destroy: ()=> void
```

```
billboard.destroy() // этот метод вызовет отписку от всех событий/слушателей и удалит баннер
```

# Формат Flowbutton

Формат представляет из себя контейнер, зафиксированный относительно viewport'a, отображающийся поверх основного контента и содержащий в себе один или два креатива, которые могут сменять друг друга по клику либо автоматически.

## Схема:

```
/**
 * Настройки кампании
 */
interface Campaign {
  p1: string;
  p2: string;
}

/**
 * Настройки формата
 */
interface FlowbuttonOptions {
  adfox: {
    ownerId: number;
    params: Campaign;
  };
}

/**
 * Публичный контроллер формата
 */
interface FlowbuttonPublicController {
  destroy: () => void;
}

/**
 * Интерфейс вызова формата
 */
interface Window {
  МТТ: {
    flowbutton: (options: FlowbuttonOptions) => FlowbuttonPublicController;
  };
}
```

## Шаблоны:

Flowbutton работает со следующими типами шаблонов

html5 - iframe

image – картинка

# Формат Fullscreen

Баннер имеет несколько более сложную структуру, нежели остальные, так как он запускается в несколько этапов:

`window.MTT.fullscreen` – задает рекламные настройки

Обратите внимание вызов `.run()` в конце на схеме, данный метод запускает работу ФС.

Необходимость усложненной схемы была выведена из следующих требований к ФС:

Необходимо снаружи реагировать на результат работы рекламного сервера (STUB, ERROR, TIMEOUT)

Предусмотреть (в будущем) возможность вызова разных рекламных серверов параллельно (Adx, Adfox) без изменения API

Обработать потенциально разные параметры и события баннера в зависимости от рекламного сервера и типа шаблона (наш, adfox)

При показе нашего типа шаблона на самом деле могут быть показаны два типа нашего шаблона (`html5`, `image_smartphone`)

## Схема:

```
// Описание типа "Адаптивные настройки". Представляет собой объект с полями, mobile, tablet, desktop,  
в которых лежат настройки <T>, настройки будут выбраны в зависимости от устройства.  
type DeviceDepends<T>;
```

```
// Описание типа "ставка". Массив таких ставок передается непосредственно в  
window.Ya.headerBidding.pushAdUnits(units);  
https://sites.help.adfox.ru/page/217  
type AdUnit = {  
  code?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}>  
}
```

```
// Упрощенный пример вызова ФС  
window.MTT.fullscreen{
```

```
  // Настройки adfox - такие же как и везде  
  adfox: {  
    ownerId: number,  
    params: AdFox.Campaign,  
    paramsAdaptive: DeviceDepends<AdFox.Campaign>  
  },
```

```
  // Unit-ы, обычные или адаптивные, для pushAdUnits  
  adfoxHeaderBidding: {  
    units: AdUnit[],  
    unitsAdaptive: DeviceDepends<AdUnit[]>  
  },
```

```
  // Необязательный элемент: селектор элемента, либо сам элемент  
  container?: string | HTML<Element>,
```

```
  // Настройки частоты показа баннера
```

```
frequency?: {  
  period: number,  
  qty: number  
}  
}).run();
```

### Упрощенный пример с контроллером ФС:

Создаем экземпляр внешнего контроллера

```
const AdObserver = window.MTT.fullscreen({...});
```

```
// Вызывается при ERROR, STUB или TIMEOUT  
AdObserver.onEmpty(function() {  
  console.log('Empty response');  
});
```

```
// Запускаем работу ФС  
AdObserver.run();
```

### Настройки рекламы детально:

#### adfox

Настройки adfox, такие же как и везде

#### adfoxHBUnits

Обычные и адаптивные настройки unit-ов для pushAdUnits

#### container?

Контейнер, куда будем вставлять ФС. По умолчанию: document.body

#### frequency?

Настройки частоты показа баннера. По умолчанию: отключено.

Настройка данного параметра ставит на страницу куку \_mtt\_fs\_s.

Свойство	Тип	Описание
period	number > 0	Период (в часах) на который ставим куку
qty	number > 0	Количество показов на этот период

#### skipButtonTimer?

Настройки кнопки закрытия fullscreen.

```
window.MTT.fullscreen({ ... })  
  .run({  
    banner: {  
      skipButtonTimer: 0, // значение 0 задаётся для мгновенной отрисовки кнопки закрытия  
    }  
  });
```

# Формат Infinite Inread

Infinite Inread рассчитан на то, что при infinite-скролле на страницу добавляются блоки, соответствующие определенному селектору.

Раз в секунду, модуль проверяет, не добавилось ли новых "страниц" по селектору `selector`.

Если добавляются, то в контексте этих страниц запускается баннер инрида так, как будто кроме этой страницы ничего не существует

## Схема:

```
window.MTT.inreadInfiniteScroll({  
  selector: string, // document.querySelectorAll селектор  
  inreadOptions: InreadOptions // опции inread  
});
```

## Структуры верхнего уровня:

### **selector**

строка, передаваемая в `document.querySelectorAll`

# Формат Inread

Поиск мест для вставки происходит следующим образом:

Берем всех первых потомков элемента-контейнера

Пробегаем по ним и отбрасываем потомков, в которых найдены "грязные элементы" (медиа-элементы, iframe, картинки)

Пробегаем по оставшимся

Складываем высоты соседних элементов, и пробуем рассчитать можно ли между "сегментами" из соседних блоков вставить блок с инридом.

Если встречаем "грязный элемент", то начинаем подсчет с нуля

Если элемент вставлен, начинаем новый "сегмент"

Все это делается в соответствии со всеми настройками отступов

## Схема:

```
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
type MaxAdsControl = {  
  count: number,  
  logBase?: number  
}
```

```
type AdUnit = {  
  code?: string,  
  codeType?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}
```

```
enum BannerResult {  
  ERROR = "ERROR",  
  LOADED = "LOADED",  
  STUB = "STUB",  
  TIMEOUT = "TIMEOUT",  
}
```

```
type AdProviderResultCallback = (isSuccess: BannerResult) => void;
```

```
type AdProviderCallback = (  
  controller: InreadPublicBlockController, // Текущий контейнер  
  blockIndex: number, // Индекс текущего контейнера  
  resultCallback: AdProviderResultCallback // cb  
) => void;
```

```
window.MTT.inread({  
  adfox: {
```

```

    ownerId: number
    params?: Campaign
    paramsAdaptive?: DeviceDepends<Campaign> // Настройки кампании в зависимости от устройства
  },
  banner: {
    container: 'selector' | HTMLElement, // Контейнер, где будет произведен поиск мест для вставки
    блоков инрида
    minScreenLength?: number, // Минимальная длина статьи (в экранах), для которой будет создаваться
    инрид
    injectionFrequency?: number, // Частота вставки инрида
    insertionPadding?: number, // Отступ от "грязных" элементов
    checkCollisions?: boolean, // Проверка на обтекание контейнера другими элементами, внутри
    которых могут быть грязные элементы
    overrideAdSizes?: DeviceDepends<number>, // Переопределение размера контейнера инрида
    topIndent?: number, // Отступ от верхнего края статьи
    lazyLoad?: boolean, // *Ленивая загрузка* баннера при приближении баннерного места к вьюпорту
    lazyLoadDistance?: number, // *Ленивая загрузка* которая происходит на заданной дистанции до
    баннерного места
    maxAdsControl?: number | MaxAdsControl // Настройки ограничения количества баннеров
  },
  adfoxHeaderBidding?: {
    units: AdUnit[],
  },
  adProvider?: AdProviderCallback,
});

```

## Основные свойства:

### adfox

настройки рекламной компании (общее для всех баннеров)

### banner

настройки баннера inread

### adfoxHeaderBidding?

Массив ставок передающихся в `window.Ya.headerBidding.pushAdUnits(units);`  
<https://yandex.ru/support/adfox-sites/monetization/header-bidding.html>

### adProvider?

Функция для работы с рекламными блоками.

Добавляет возможность сначала сделать запрос стороннего вендора (Native-roll, VideoNow, итд),

и в случае отсутствия в них рекламы запросить адфокс.

Получает контейнер рекламного места, его индекс, и коллбэк.

## Настройки баннера:

### banner.container

Контейнер, где будет произведен поиск мест для вставки блоков инрида.

Ожидает строку selector для `document.querySelector` или HTML – элемент.

### banner.minScreenLength?

Минимальная длина статьи (в экранах), для которой будет создаваться инрид.

По умолчанию 0.1

#### **banner.injectionFrequency?**

Частота вставки инрида. Не более 1 баннера на N экранов пользователей.

По умолчанию 0.1

#### **banner.checkCollisions?**

Проверка обтекания контейнера, другими элементами. Обтекающие элементы уменьшают визуальную ширину статьи, при этом (сюрприз!) программная ширина останется старой

Пример: [https://www.kinonews.ru/article\\_84215](https://www.kinonews.ru/article_84215)

Не стоит включать этот параметр без необходимости.

По умолчанию: false

#### **banner.overrideAdSizes?**

Переопределение размера контейнера инрида. Можно определить свой размер для каждого вида устройств.

По умолчанию для всех устройств: 300px

#### **banner.lazyLoad?**

Lazyload для inread

По умолчанию: true

#### **banner.lazyLoadDistance?**

Кастомное значение для Lazyload в inread, задается в пикселях.

#### **banner.topIndent?**

Размер отступа от верхнего края статьи для первого инрида в процентах.

По умолчанию: 0

#### **banner.maxAdsControl?**

Ограничение количества баннеров в блоке.

При ограничении количества баннеров, отключать излишек необходимо как можно более равномерно, а не с начала статьи.

Делается это с помощью логарифма, для примера попробуйте поставить 2.2 на статье из 14-16-ти баннеров.

Можно передать два типа значений:

число – максимальное количество баннеров

объект MaxAdsControl

Свойство	Тип	Описание
count	number	Максимальное количество баннеров
logBase?	number	Основание логарифма, должно быть больше 1, может быть дробным. По умолчанию – 2

# Формат Interscroller

Формат встраивается в контент сайта, визуально "разрывая" его. Отображается так, как будто это подложка на нижнем слое, которая при скролле фиксируется на месте

## Схема:

```
// Описание типа "Рекламная кампания"  
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
// Описание типа "Адаптивные настройки". Представляет собой объект с полями, mobile, tablet, desktop,  
в которых лежат настройки <T>, настройки будут выбраны в зависимости от устройства.  
type DeviceDepends<T>;
```

```
// Описание типа "ставка". Массив таких ставок передается непосредственно в  
window.Ya.headerBidding.pushAdUnits(units); // https://sites.help.adfox.ru/page/217  
type AdUnit = {  
  code?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}
```

```
// Описание параметров формата  
interface InterscrollerParams {
```

```
  // Настройки adfox - такие же как и везде  
  adfox: {  
    ownerId: number,  
    params: Campaign,  
    paramsAdaptive?: DeviceDepends<Campaign>  
  },
```

```
  // Unit-ы, обычные или адаптивные, для pushAdUnits  
  adfoxHeaderBidding?: {  
    units?: AdUnit[],  
    unitsAdaptive?: DeviceDepends<AdUnit[]>  
  },
```

```
  // Настройки формата  
  banner: {  
    // селектор элемента, либо сам элемент  
    container: string | HTMLElement;
```

```
  // позиция вставки  
  insertPosition?: InsertPosition;
```

```
  // высота "окна", в котором отображается баннер  
  height?: string | DeviceDepends<string>;
```

```
  // ленивая загрузка
```

```
lazyload?: boolean;  
  
// блокировка скролла (ms) в момент прохождения баннера через выюпорт  
disableScrollMs?: number;  
  
// лейблы "реклама" на местах "разрыва" страницы  
label?: boolean;  
  
// фон для всего баннерного места в формате HEX  
background?: string;  
}  
}
```

## Структуры верхнего уровня:

### **adfox**

настройки рекламной компании (общее для всех баннеров)

### **adfoxHeaderBidding?**

Массив ставок передающихся в `window.Ya.headerBidding.pushAdUnits(units)`;  
<https://yandex.ru/support/adfox-sites/monetization/header-bidding.html>

## Настройки баннера:

### **banner.container**

селектор элемента, либо элемент, относительно которого будет выполнена вставка баннера

### **banner.insertPosition?**

позиция вставки баннера относительно контейнера; возможные варианты: `beforebegin`, `afterbegin` (по дефолту), `beforeend`, `afterend`;  
<https://developer.mozilla.org/ru/docs/Web/API/Element/insertAdjacentHTML>

### **banner.height?**

высота "окна", в котором отображается баннер;  
примеры: `'100vh'`, `'500px'`, `'200pt'` и т.д.;  
не используйте % в качестве единиц измерения

### **banner.lazyload?**

ленивая загрузка; если включена, загрузка начнётся за одну высоту выюпорта до баннера

### **banner.disableScrollMs?**

время в миллисекундах, на которое будет заблокирован скроллинг страницы при прохождении баннера через viewport

### **banner.label?**

полосы с текстом сверху и снизу на разрывах страницы для выделения баннера

### **banner.background?**

фон для выделения баннера

**Destroy:**

В версии 1.8.0 добавлена возможность дестроя баннера снаружи.

Это необходимо для корректной работы в рамках спа-приложений.

```
const interscroller = window.MTT.interscroller({...}) // interscroller вернет публик-контроллер который имеет метод destroy: ()=> void
```

```
interscroller.destroy() // этот метод вызовет отписку от всех событий/слушателей и удалит баннер
```

# Формат Wrapper

Wrapper – это AdFox-обертка для элемента, предоставляющая набор инструментов для работы с баннерами.

Сам wrapper не создает никаких элементов, для работы ему нужны настройки adfox и id элемента-контейнера.

## Схема:

```
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
// Описание типа "Адаптивные настройки". Представляет собой объект с полями, mobile, tablet, desktop,  
в которых лежат настройки <T>, настройки будут выбраны в зависимости от устройства.  
type DeviceDepends<T>;
```

```
// Описание типа "ставка". Массив таких ставок передается непосредственно в  
window.Ya.headerBidding.pushAdUnits(units);
```

```
// https://sites.help.adfox.ru/page/217
```

```
type AdUnit = {  
  code?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}
```

```
window.MTT.wrapper({  
  // Настройки adfox - такие же как и везде  
  adfox: {  
    ownerId: number  
    params?: Campaign  
    paramsAdaptive?: DeviceDepends<Campaign> // Настройки кампании в зависимости от устройства  
  },
```

```
  // Unit-ы, обычные или адаптивные, для pushAdUnits  
  adfoxHeaderBidding?: {  
    units: AdUnit[],  
    unitsAdaptive: DeviceDepends<AdUnit[]>  
  },
```

```
  // Общие настройки  
  banner?: {  
    lazyLoad?: {  
      distance: number  
    },  
    refresh?: {  
      checkVisibility?: boolean,  
      seconds: number  
    }  
  },
```

```
containerId: string, // ID элемента
});
```

## Структуры верхнего уровня:

### **adfox**

настройки рекламной компании (общее для всех баннеров)

### **banner?**

настройки баннера, содержит набор инструментов

### **adfoxHeaderBidding?**

настройки рекламной компании (общее для всех баннеров)

### **containerId**

ID элемента строкой (без #)

## Настройки баннера:

### **banner.lazyLoad?**

Объект. Включает ленивую загрузку баннера.

Есть обязательный параметр `distance` – положительное число либо 0. Отвечает за расстояние от целевого элемента до верха или низа viewport-а браузера, на котором начнется загрузка баннера.

Примечание: не использовать `distance: 0` на блоках в первом экране, блок периодически не загружается. Лучше отказаться от использования параметра `distance` на блоках в первом экране.

Например, при скролле страницы вниз, загрузка баннера начнется при расстоянии в 300 пикселей от нижнего края страницы.

### **banner.refresh?**

Объект. Включает перезагрузку баннера методом `reload`.

Вызов перезагрузки производится с помощью `setInterval`.

<code>checkVisibility?</code>	boolean	Если true, при вызове проверяется что баннер на 100% во вьюпорте.
<code>seconds?</code>	number	Время вызова перезагрузки <b>в секундах</b> . По умолчанию - 30.

# Работа с Header Bidding

Для работы с AdFox Header Bidding его необходимо проинициализировать.  
Делается это один раз на странице

```
// Параметры описаны на странице https://sites.help.adfox.ru/page/217  
// AdUnits будут переданы позже.  
window.MTT.headerBidding({  
  adfoxBiddersMap: AdfoxBiddersMap, // Bidders Map  
  timeout: number  
});
```

Затем AdUnits передаются в баннер.

На данный момент поддерживаются FullScreen и Inread.

# Формат Sticker

## Схема:

```
type Campaign = {  
  p1: string,  
  p2: string  
}
```

```
type AdUnit = {  
  code?: string,  
  sizes: Array<[number, number]>,  
  bids: Array<{  
    bidder: string,  
    params: {  
      placementId: string  
    }  
  }>  
}
```

```
type Position = {  
  "bottom-left": "bottom left",  
  "bottom-right": "bottom right",  
  "top-left": "bottom left",  
  "top-right": "bottom right",  
}
```

```
window.MTT.sticker({  
  adfox: {  
    ownerId: number,  
    params?: Campaign,  
    paramsAdaptive?: DeviceDepends<Campaign>, // Настройки кампании в зависимости от устройства  
  },  
  adfoxHeaderBidding?: {  
    units: AdUnit[],  
  },  
  banner?: {  
    skipButtonTimer?: number; // Таймер отсчёта до появления крестика  
    position?: Position; // Определяет позицию для фикса баннера: прокидываем строку "bottom-left" ||  
"bottom-right" || "top-right" || "top-left"  
    reloadDelay?: number; // Задержка до перезагрузки баннера  
    maxReloads?: number; // Максимальное количество перезагрузок баннера  
    refresh?: number; // Время по истечении которого будет произведен рефреш баннера. В секундах.  
  },  
});
```

## Структуры верхнего уровня:

### adfox

Настройки рекламной компании (общее для всех баннеров)

### adfoxHeaderBidding?

Массив ставок передающихся в `window.Ya.headerBidding.pushAdUnits(units)`;

<https://yandex.ru/support/adfox-sites/monetization/header-bidding.html>

## **banner?**

Настройки баннера sticker

### **Настройки баннера:**

#### **banner.position?**

Настройка позволяет выбрать где должен зафиксироваться баннер.

Первым пишется положение по оси y (top или bottom) через тире ось x (left или right)

По умолчанию: "bottom left" (нижний левый край)

#### **banner.reloadDelay?**

Настройка позволяет включить перезагрузку баннера. В секундах. По дефолту отключена.

#### **banner.maxReloads?**

Настройка позволяет включить перезагрузку баннера.

Определяет количество перезагрузок которые может сделать баннер.

#### **banner.skipButtonTimer?**

Настройка позволяет включить отображение обратного отсчёта до появления крестика. В секундах. По дефолту крестик виден сразу.

#### **banner.refresh?**

Настройка позволяет включить рефреш баннера. В секундах. Менее 30 сек ставить бессмысленно, AdFox обновляет не чаще 1 раза в 30 сек

#### **Destroy:**

```
const sticker = window.MTT.sticker({...}) // sticker вернет публик-контроллер который имеет метод destroy:  
(=> void  
sticker.destroy() // этот метод вызовет отписку от всех событий/слушателей и удалит баннер
```