jsAPI для XRAD

Выпуск 6.6.7

Хи-Квадрат

июл. 10, 2026

Содержание

ГЛАВА 1

Описание

Объект jsAPI, доступный в любой части Вашего javascript-кода - это основной интерфейс для создания интерактивных приложений и взаимодействия с сервером.

Он предоставляет гибкие и надежные методы для отображения диалоговых окон, взаимодействия с свойствами компонентов и выполнения сетевых запросов.

jsAPI включен в систему по умолчанию и находится в глобальной области видимости PGHS (window).

ГЛАВА 2

component(...)

`jsAPI.component(id: string): Object`

Основной интерфейс доступа к методам и свойствам компонентов.

Аргумент	Тип	По умолчанию	Описание
`id` *	`string`	–	идентификатор компонента

Возвращает: Объект с методами и свойствами, доступными для компонента с указанным `id`

(!) Методы и свойства указаны для каждого компонента в блоке с его описанием.

(!) Если вы попытаетесь обратиться к несуществующему свойству или вызвать несуществующий метод - это не вызовет критической ошибки, но в консоль будет выведено соответствующее сообщение.

(!) Аргументы методов с одинаковым названием для различных компонентов могут отличаться. Внимательно изучите раздел документации, описывающий методы и аргументы соответствующего компонента.

2.1 Кнопка (BUTTON)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

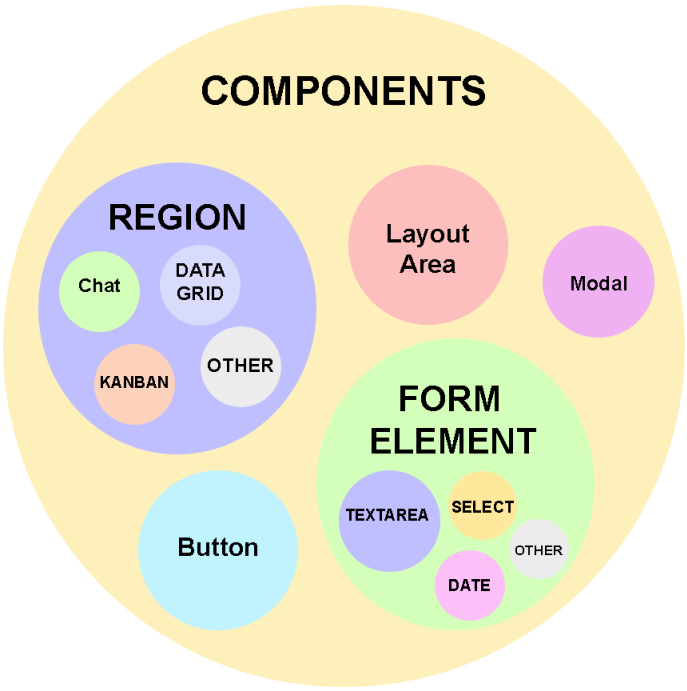

Рис. 1: Визуализация структуры компонентов в PGHS

Свойство	Тип	Описание
`attributes`	`object`	Отображать состояние загрузки?
`attributes.class`	`string`	CSS-классы для кнопки
`title`	`string`	Текст, отображаемый при наведении
`target`	`string`	Атрибут target для ссылки
`buttonID` readonly	`string`	Идентификатор кнопки
`loading`	`boolean`	Отображать состояние загрузки?
`class`	`string`	CSS-классы для кнопки
`col` readonly	`number`	Количество колонок, которые занимает регион 1-12
`href`	`string`	Ссылка для перехода при нажатии
`list` readonly	`Array<IDropdownNavMenuItem>`	Список, разворачиваемый при нажатии на кнопку
`list.current`	`boolean`	Выделяет элемент списка как активный
`list.href`	`string`	Ссылка, на которую следует перейти по нажатию на элемент списка
`list.icon`	`string`	CSS-класс иконки
`list.name`	`string`	Текст, выводимый внутри элемента списка
`list.separated`	`boolean`	Отделен ли элемент списка визуально от остальных? (обычно полосой)

продолжается на следующей странице

Таблица 2 – продолжение с предыдущей страницы

Свойство	Тип	Описание
`list.tooltip`	`string`	Текст, выводимый при наведении на элемент списка
`position`	`top-right` | `top-left` | `bottom-right` | `bottom-left` | `bottom-fluid` | `right-body` | `left-body`	Положение кнопки в родительском регионе
`request`	`string`	Запрос, который следует выполнить при нажатии на кнопку
`setLoading`	`boolean`	Следует ли установить состояние загрузки на время выполнения запроса? (работает только если указан `request`)
`list.skipValidations`	`boolean`	Следует ли пропускать клиентские валидации при нажати на кнопку?
`text`	`string`	Текст, отображаемый внутри кнопки

2.2 Регионы

Свойства, доступные для всех регионов

Вы можете изменять значения этих свойств в реальном времени при помощи метода `jsAPI.component(...).properties(...)` (см. ниже).

Свойство	Тип	Описание
`id` readonly	`string`	Идентификатор региона
`uid` readonly	`string`	Уникальный идентификатор региона
`active` not recommended	`boolean`	Разрешено ли сворачивание региона?
`title`	`string`	Текст в заголовке
`class`	`string`	CSS-класс региона
`componentClass`	`string`	CSS-класс компонента
`col` readonly	`number`	Количество колонок, которые занимает регион 1-12
`component` readonly	`string`	Тип региона
`childrens` readonly	`Array<IComponent>`	Дочерние регионы
`data` not recommended	`Object`	Данные, специфичные для конкретного региона
`settings`	`Object`	Настройки отображения
`settings.background`	`boolean`	Имеет ли регион фон?
`settings.border`	`boolean`	Имеет ли регион обводку?
`settings.boxShadow`	`boolean`	Имеет ли регион внешнее свечение?
`settings.containerCentered`	`boolean`	Центрирован ли регион по горизонтали?
`settings.containerWidth`	`number`	Максимальная ширина региона

продолжается на следующей странице

Таблица 3 – продолжение с предыдущей страницы

Свойство	Тип	Описание
`settings.footerSeparated`	`boolean`	Подвал отделен? (если есть)
`settings.headerSeparated`	`boolean`	Шапка отделена? (если есть)
`settings.padding`	`boolean`	Имеет ли регион внутренние отступы?
`showHeader`	`boolean`	Необходимо ли отображать шапку?
`toggle`	`boolean`	Разрешено ли сворачивание региона? (только если `showHeader === true`)

2.2.1 Хлебные крошки (BREADCRUMB)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент `options.rerender`	Тип `Boolean`	По умолчанию `false`	Описание Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)

продолжается на следующей странице

Таблица 4 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options. onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

2.2.2 Карточки (CARDS)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options. changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 5 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.3 График (CHART)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 7 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.4 Чат (CHAT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 9 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.5 Дата-грид (DATAGRID)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент `options.rerender`	Тип `Boolean`	По умолчанию `false`	Описание Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 11 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.6 Календарь (CALENDAR)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 13 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.7 HTML (HTML_TEXT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 15 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.8 Kanban-доска (KANBAN)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 17 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

2.2.9 Навигационная панель (PAGE NAVIGATION)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;

- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

2.2.10 Отчет (REPORT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 19 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.11 Таблица (TABLE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 21 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

2.2.12 Вкладки (TABS)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;

- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

- `jsAPI.component(...).changeTab(tabNumber: number) : void` - **Переключить** активную вкладку;

- `jsAPI.component(...).hideTab(tabNumber: number) : void` - **Скрыть** вкладку.
  Если вкладка активна, то после скрытия автоматически будет активирована ближайшая доступная вкладка;

- `jsAPI.component(...).showTab(tabNumber: number) : void` - **Отобразить** ранее скрытую вкладку.
  Если не выбрана никакая вкладка, то отображенная вкладка будет активирована автоматически;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.13 Плитки (TILES)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 24 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.14 Дерево (TREE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 26 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.15 Визард (WIZARD)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).refresh(options: IRefreshOptions): void` - Обновляет регион с учётом указанных параметров (items);

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.changeState`	`Boolean`	`true`	Сменять ли состояние компонента автоматически?

продолжается на следующей странице

Таблица 28 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.rerender`	`Boolean`	`false`	Необходимо ли выполнить перерисовку компонента после получения ответа от сервера?
`options.items`	`Record<string, string>`	`null`	Параметры для отправки на сервер. При указании этой опции содержимое `items` будет полностью переопределено перед отправкой на сервер. Поддерживаются только значения в формате `string`. Пример: `{ 'client': '48561' }`
`options.sendData`	`Record<string, unknown>`	`null`	Данные для отправки на сервер (полностью переопределяют `payload`, использовать с осторожностью)
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

2.2.16 Обертка (WRAPPER)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?

2.2.17 Форма (APP_FORM)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;

Таблица 30: Свойства компонента APP_FORM

Свойство	Тип	Описание
`loading`	`boolean`	Отображать состояние загрузки?
`data.items` readonly	`Array<IRegionField>`	Массив с элементами формы

2.3 Элементы формы

Свойства, доступные для всех элементов формы

Вы можете изменять значения этих свойств в реальном времени при помощи метода `jsAPI.component(...).properties(...)`.

Свойство	Тип	Описание
`col`	`number`	Количество колонок, которые занимает элемент формы 1-12
`cssClass`	`string`	CSS-класс, который будет добавлен на корневой элемент
`disabled`	`boolean`	Отключено
`label`	`string`	Текстовая подпись элемента формы, которая видна пользователю
`placeholder` устарело	`string`	~~Текстовая подпись элемента формы, которая видна пользователю~~
`postText`	`string`	Текст или HTML, выводимый после поля ввода
`preText`	`string`	Текст или HTML, выводимый перед полем ввода

продолжается на следующей странице

Таблица 31 – продолжение с предыдущей страницы

Свойство	Тип	Описание
`required`	`boolean`	Обязательно ли поле для заполнения. Проверка по этому параметру выполняется только на стороне клиента. **(!) Изменение этого параметра не влияет на серверную валидацию.**
`visuallyRequired`	`boolean`	**Визуальное отображение** обязательности поля. Не влияет на валидацию поля
`readonly` устарело	`boolean`	~~Доступно только для чтения~~
`tooltip`	`string`	Подсказка для поля ввода
`trim`	`NONE` | `LEADING` | `TRAILING` | `BOTH`	Нужно ли удалять пробелы у введенного значения? `NONE` - не очищать, `LEADING` - слева, `TRAILING` - справа, `BOTH` - слева и справа
`type` readonly	`string`	Тип поля ввода (`TEXT`, `RADIOS`, `NUMBER`...)
`jsSettings` readonly	`Object`	Объект с настройками для элемента формы. Имеет структуру, специфичную для каждого компонента

2.3.1 Автозаполнение (AUTOCOMPLETE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.2 Флажки (CHECKBOXES)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus(value: string = undefined) : void` - сфокусировать курсор на опции выбора;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`list`	`Array<ISelectListOption>`	Список опций для выпадающего списка
`list[].display`	`string`	Отображаемый текст для опции
`list[].value`	`string`	Значение, которое будет установлено при выборе опции

2.3.3 Дата (DATE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.4 Выбор файла (FILE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.5 Множественный выбор (MULTISELECT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus(open: boolean = true) : void` - сфокусировать курсор на опции выбора;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`list`	`Array<ISelectListOption>`	Список опций для выпадающего списка
`list[].display`	`string`	Отображаемый текст для опции

продолжается на следующей странице

Таблица 33 – продолжение с предыдущей страницы

Свойство	Тип	Описание
`list[].value`	`string`	Значение, которое будет установлено при выборе опции

2.3.6 Числовой (NUMBER)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.7 Радиокнопки (RADIOS)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus(value: string = undefined) : void` - сфокусировать курсор на опции выбора;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`list`	`Array<ISelectListOption>`	Список опций для выпадающего списка
`list[].display`	`string`	Отображаемый текст для опции
`list[].value`	`string`	Значение, которое будет установлено при выборе опции

2.3.8 Список (SELECT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus(open: boolean = true) : void` - сфокусировать курсор на опции выбора;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`list`	`Array<ISelectListOption>`	Список опций для выпадающего списка
`list[].display`	`string`	Отображаемый текст для опции
`list[].value`	`string`	Значение, которое будет установлено при выборе опции

2.3.9 Переключатель (SWITCHER)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus(value: string = undefined) : void` - сфокусировать курсор на опции выбора;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`list`	`Array<ISelectListOption>`	Список опций для выпадающего списка
`list[].display`	`string`	Отображаемый текст для опции
`list[].value`	`string`	Значение, которое будет установлено при выборе опции

2.3.10 Пароль (PASSWORD)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.11 Текст (TEXT)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;
- `jsAPI.component(...).setMask(...args: InputmaskArgs): void` - установить маску для поля ввода.

Аргумент	Тип	По умолчанию	Описание
`...args`	`[string] | [string, Record<string, any>] | [Record<string, any>] | any[]`	–	Маска для поля ввода. С поддерживаемым форматом маски можно ознакомиться в официальной документации inputmask

- `jsAPI.component(...).removeMask(): void` - удалить маску с поля ввода.

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`mask` readonly	`string`	Текущая маска для поля ввода
`maxlength`	`number`	Максимальное количество символов для поля ввода

2.3.12 Текстовая область (TEXTAREA)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

Свойства (properties), доступные для чтения при помощи `jsAPI.component(...).info` и редактирования при помощи `jsAPI.component(...).properties(...)`:

Свойство	Тип	Описание
`maxlength`	`number`	Максимальное количество символов для поля ввода

2.3.13 Переключатель (TOGGLE)

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

2.3.14 WYSIWYG

- `jsAPI.component(...).rerender() : void` - *выполнить повторную отрисовку компонента*;
- `jsAPI.component(...).info: Object` - *актуальные свойства компонента*;
- `jsAPI.component(...).properties(props: Record<string, string | number>) : void` - *обновить свойства компонента*;
- `jsAPI.component(...).focus() : void` - сфокусировать курсор на компоненте;

ГЛАВА 3

:colorScheme

`jsAPI.colorScheme`

Объект доступа к функциям смены цветовой схемы (если она была настроена).
Методы этого раздела позволяют управлять цветовой схемой приложения.
Все методы управления цветовой схемой взаимодействуют лишь с текущим устройством пользователя и не используют сервер для хранения состояний.
По умолчанию на устройстве пользователя будет включена цветовая схема, которая выбрана как приоритетная на устройстве пользователя.

Вы можете принудительно включить нужную цветовую схему, либо использовать методы `jsAPI.colorScheme` для добавления опции выбора цветовой схемы в вашем приложении.

Смена цветовой схемы реализована посредством переключения CSS-класса на корневом HTML-элементе. `.--theme-light`, `.--theme-dark`

3.1 colorScheme.toggle()

```
jsAPI.colorScheme.toggle() : void
```

Переключить цветовую схему на текущем устройстве пользователя.

Текущая темная -> включится светлая
Текущая светлая -> включится темная

(!) Вам достаточно лишь добавить пункт в навигационную панель вашего приложения и вызвать этот метод для базовой интеграции переключения цветовой схемы.

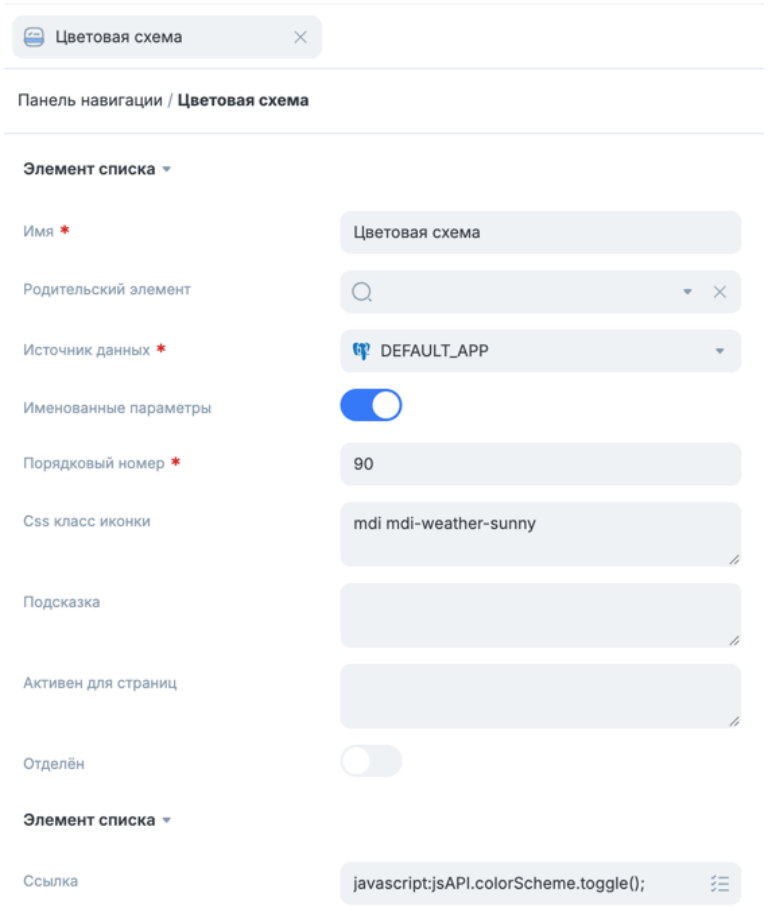

Рис. 1: Пример реализации переключения цветовой схемы в панели навигации

3.2 colorScheme.dark()

`jsAPI.colorScheme.dark() : void`

Принудительно включить **темную** тему на текущем устройстве пользователя.

3.3 colorScheme.light()

`jsAPI.colorScheme.light() : void`

Принудительно включить **светлую** тему на текущем устройстве пользователя.

3.4 colorScheme.reset()

`jsAPI.colorScheme.reset() : void`

Вернуть цветовую схему к той, которая выбрана как предпочитаемая на текущем устройстве пользователя

3.5 :colorScheme.info

`jsAPI.colorScheme.info: Object`

Информация о текущей цветовой схеме на устройстве пользователя

Для отслеживания актуального состояния всегда получайте значения напрямую из `jsAPI.colorScheme.info`, не записывайте ответ в переменную.

```
// Неправильно
const scheme = jsAPI.colorScheme.info;
console.log(scheme.current); // dark
jsAPI.colorScheme.light(); // включаем светлую цветовую схему
console.log(scheme.current); // dark (хотя актуальная тема - light)

// Правильно
console.log(jsAPI.colorScheme.info.current); // light
jsAPI.colorScheme.toggle(); // переключаем цветовую схему
console.log(jsAPI.colorScheme.info.current); // dark (верно)
```

Таблица 1: Содержимое `jsAPI.colorScheme.info`

Свойство	Тип	Возможные значения	Описание
`current`	`string`	`'dark'|'light'`	Текущая цветовая схема
`systemPreferredColorS`	`string`	`'dark'|'light'`	Цветовая схема, которая указана как „предпочитаемая“ на текущем устройстве пользователя

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Свойство	Тип	Возможные значения	Описание
`userSelectedColorSche`	`string`	`'dark'|'light'`	Цветовая схема, которая была установлена на устройстве пользователя принудительно (при помощи методов `jsAPI.colorScheme.light()`, `jsAPI.colorScheme.dark()`, `jsAPI.colorScheme.toggle()`)
`isUserSelected`	`boolean`	`true|false`	Была ли текущая тема выбрана пользователем?
`isSystemPreference`	`boolean`	`true|false`	Текущая тема выбрана на основе системных предпочтений?

ГЛАВА 4

:date

`jsAPI.date`

Модуль для работы с датой.

4.1 jsAPI.date.getCurrent(...)

`jsAPI.date.getCurrent(offset: string) : string`

Получить дату в формате «`DD.MM.YYYY`» со сдвигом относительно текущей даты на устройстве пользователя.

Таблица 1: Возможные значения `offset`

Символ	Значение
`-`	Отнять от текущей даты
`+`	Прибавить к текущей дате
`d` | `D`	Сдвиг на несколько дней
`w` | `W`	Сдвиг на несколько недель
`m` | `M`	Сдвиг на несколько месяцев
`y` | `y`	Сдвиг на несколько лет

Формат: `/^(-|\+)([1-9][0-9]{0,3})[dmyw]$/;`

```
// Добавить одну неделю к текущей дате
jsAPI.date.getCurrent("+1w");

// Добавить две недели к текущей дате
jsAPI.date.getCurrent("+2w");

// Отнять две недели от текущей даты
jsAPI.date.getCurrent("-2w");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
// Отнять один год и два месяца от текущей даты
jsAPI.date.getCurrent("-1y -2m");

// Отнять один год, два месяца и четыре дня от текущей даты
jsAPI.date.getCurrent("-1y -2m -4d");

// Может быть полезно для формирования ограничений по возрасту - совершеннолетие
console.log("Для заключения договора вы должны быть рождены не позднее " + jsAPI.date.
↪getCurrent("-18y"));
```

ГЛАВА 5

download(...)

```
jsAPI.download(options: IDownloadRequestOptions) : Promise<IDownloadFileResult>
```

Скачать файл через JavaScript, без перенаправления или обновления страницы.
Этот метод позволяет отслеживать прогресс скачивания файла и обрабатывать ошибки возникшие во время генерации.
Вы можете заблокировать кнопку генерации файла или вывести информацию о прогрессе пользователю.
Это может быть особенно полезно если генерация файла занимает длительное время.

Таблица 1: `IDownloadRequestOptions`

Аргумент	Тип	По умолчанию	Описание
`process`	`string`	–	Имя процесса, который необходимо вызвать для скачивания файла
`url`	`string`	–	URL для скачивания файла (файл или метод API)
`method`	`GET | POST`	–	Метод запроса
`data`	`Record<string, any> | FormData`	–	данные, которые необходимо отправить. Для метода POST реализована поддержка формата данных *FormData*
`headers`	`Record<string, string>`	–	заголовки запроса
`onProgress`	`(progress: ProgressEvent) => void`	–	Коллбэк, вызываемый при обновлении прогресса скачивания файла

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`onSuccess`	`(result: IDownloadFileResult) => void`	–	Коллбэк, вызываемый при успешном скачивании файла
`onError`	`(error: IDownloadFileResult[e => void`	–	Коллбэк, вызываемый при возникновении ошибки во время скачивания файла файла

`ProgressEvent` (документация: https://developer.mozilla.org/en-US/docs/Web/API/ProgressEvent)

Таблица 2: `IDownloadFileResult`

Свойство	Тип	Описание
`success`	`boolean`	Успешность операции
`response`	`any`	Ответ от сервера
`error.title`	`string`	Заголовок ошибки (если есть)
`error.message`	`string`	Сообщение ошибки (если есть)
`error.code`	`string | number`	Код ошибки (если есть)

Таблица 3: Коды ошибок

Код	Описание
-2	Ошибка интернет-соединения
4xx,5xx	сервер вернул ошибку

```
// Скачать файл сгенерированный процессом `fooProcess` с данными из полей `USERNAME`␣
↪и `AGE`
jsAPI.download({
  process: `fooProcess`,
  data: {
    username: jsAPI.getItem('USERNAME'),
    age: jsAPI.getItem('AGE'),
  }
})
```

```
// Скачать файл /test.txt и отобразить прогресс
jsAPI.download({
  url: "/test.txt",
  method: "GET",
  onSuccess: function() {
    jsAPI.notification.success("Скачивание завершено!");
  },
  onError: function(error) {
    console.error(error)
    jsAPI.notification.error("Ошибка #" + error.code + " <br>" + error.message);
  },
  onProgress: function(progress) {
    const percent = progress.loaded / progress.total * 100;
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    jsAPI.notification.warning("Скачано " + percent + "%");
  }
});
```

```
// Отправить данные методом POST в объекте `FormData` на адрес `/api/download_file` с
↪заголовком `token` и заблокировать кнопку на время скачивания:
const payload = new FormData();
payload.append('username', jsAPI.getItem('USERNAME'));
payload.append('age', jsAPI.getItem('AGE'));

jsAPI.component("BUTTON").properties({ loading: true });  // Блокируем кнопку

jsAPI.download({
  url: "/api/download_file",
  method: "POST",
  data: payload,
  headers: { token: '@JKnfsdlk@E$@!BNKOGDSbjO#$BKJO' },
  onSuccess: function() {
    jsAPI.component("BUTTON").properties({ loading: false }); // Разблокируем кнопку
    jsAPI.notification.success("Скачивание завершено!");
  },
  onError: function(error) {
    jsAPI.component("BUTTON").properties({ loading: false }); // Разблокируем кнопку
    console.error(error)
    jsAPI.notification.error("Ошибка #" + error.code + " <br>" + error.message);
  },
  onProgress: function(progress) {
      const percent = progress.loaded / progress.total * 100;
      jsAPI.notification.warning("Скачано " + percent + "%");
  }
});
```

ГЛАВА 6

getAppInfo()

`jsAPI.getAppInfo() : Object`

Возвращает информацию о сборке приложения.

Свойство	Тип	Описание
`version`	`string`	Версия приложения
`staticVersion`	`string`	Версия сборки PGHS

```
// Получить версию приложения
jsAPI.getAppInfo().version;

// Получить версию сборки PGHS
jsAPI.getAppInfo().staticVersion;
```

ГЛАВА 7

getCurrentPage()

`jsAPI.getCurrentPage() : IPageLayer`

Возвращает информацию о **текущей** странице.

Таблица 1: IPageLayer

Свойство	Тип	Описание
`clientErrors`	`Record<IRegion['id'], string>`	Отображаемые ошибки валидации
`clientValidations`	`Record<IRegion['id'], string>`	Все актуальные ошибки валидации (но не все отображаемые. Отображаемые нужно брать из `clientErrors`)
`items`	`Record<string, string | boolean | number>`	Значения для полей ввода, которые мы получили с сервера. А так же актуальные значения полей ввода, которые прошли валидацию и фильтрацию, полностью готовые к отправке на сервер.
`models`	`Record<string, string | boolean | number>`	Модель данных для отображения значений в полях ввода. Динамические значения, которые меняются во время ввода данных
`page`	`IPage`	Исходные данные страницы, которые мы получили с сервера
`pageId`	`number | string`	Сгенерированный, уникальный, внутренний идентификатор страницы

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Свойство	Тип	Описание
regionData	Record<IRegion['id'], unknown>	Вы можете использовать это свойство для передачи данных в processPage. Клиент может установить любое значение в это свойство. Это значение будет передано во время отправки processPage. Пример реализации: хранение несохраненных изменений в DataGrid. Несохраненные данные будут отправлены автоматически при срабатывании processPage.
serverErrors	Record<string, ResponseException>	Ошибки, которые мы получили с сервера
src	string	Ссылка на страницу. Обычно это номер страницы

При вызове метода при открытом модальном окне - будет получена информация о самом верхнем модальном окне

```
type ResponseException = {
    code: string;
    title: string;
    message: string;
    errors?: Array<{
        title: string;
        text: string;
        code: string;
    }>;
    redirect: IResponseRedirect;
};

interface IResponseRedirect {
    link: string;
    title?: string;
    type: RedirectType;
    width?: string;
}
```

```
// Получить id страницы
jsAPI.getCurrentPage().pageId;
```

```
// Получить значения item-а
jsAPI.getCurrentPage().items["P1000_MY_ITEM"];
```

ГЛАВА 8

setItem(...)

`jsAPI.setItem(name: string, value: string | number, options: ISetItemOptions): void`

Изменяет значение элемента формы.

Аргумент	Тип	По умолчанию	Описание
`name`	`string`	–	Имя элемента формы. НЕ является селектором (указывать без '#').
`value`	`string`	–	Новое значение для элемента формы. Пусто - очистка
`options.triggerDA`	`boolean`	`true`	Необходимо ли посылать событие `change` при изменении значения элемента формы

```
// Присвоить значение 'Иванов Иван' элементу формы с именем 'P1000_ITEM' и при этом
↪не посылать событие 'change'
jsAPI.setItem("P1000_ITEM", "Иванов Иван". { triggerDA: false });

// Присвоить значение 'Иванов Иван' элементу формы с именем 'P1000_ITEM'
jsAPI.setItem("P1000_ITEM", "Иванов Иван");

// Очистить значение элемента формы с именем 'P1000_ITEM'
jsAPI.setItem("P1000_ITEM", "");
```

ГЛАВА 9

getItem(...)

`jsAPI.getItem(name: string): string`

Возвращает значение элемента формы

Аргумент	Тип	По умолчанию	Описание
`name`	`string`	–	Идентификатор элемента формы. **НЕ является селектором (указывать без '#')**

Возвращает: `string` - значение элемента формы

```
// Получить актуальное значение поля 'P1000_ITEM'
jsAPI.getItem("P1000_ITEM");
```

ГЛАВА 10

hideItem(...)

`jsAPI.hideItem(name: string) : void`

Скрывает элемент формы (item) из DOM и привязанную к нему колонку

Аргумент	Тип	По умолчанию	Описание
`name`	`string`	–	Идентификатор элемента формы. **НЕ является селектором (указывать без '#')**

```
// Скрыть элемент формы с идентификатором 'P1000_ITEM'
jsAPI.hideItem("P1000_ITEM");
```

ГЛАВА 11

showItem(...)

`jsAPI.showItem(name: string) : void`

Возвращает скрытый при помощи *jsAPI.hideItem(…)* элемент формы в DOM и привязанную к нему колонку

Аргумент	Тип	По умолчанию	Описание
`name`	`string`	–	Идентификатор элемента формы. **НЕ является селектором (указывать без '#')**

```
// Отобразить ранее скрытый элемент формы с идентификатором 'P1000_ITEM'
jsAPI.showItem("P1000_ITEM");
```

ГЛАВА 12

logout()

`jsAPI.logout() : void`

Совершает выход из приложения (разлогин), вызывая метод `{API_URL}/logout`

```
// Разлогинить текущего пользователя из системы
jsAPI.logout();
```

ГЛАВА 13

:modal

`jsAPI.modal`

Модуль для управления модальными окнами приложения.

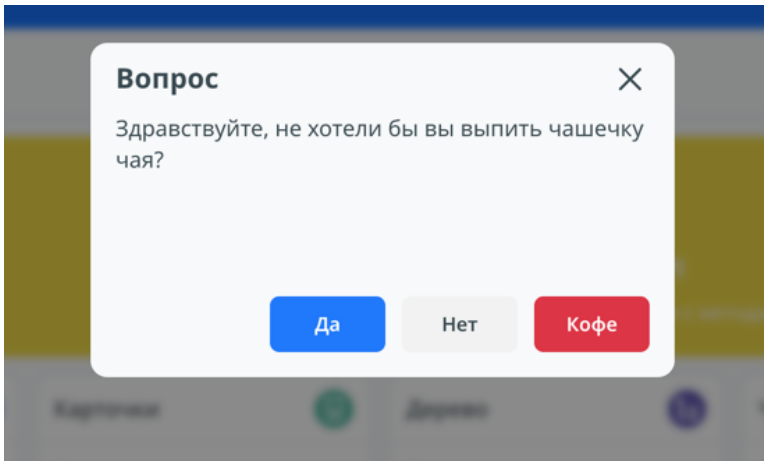

Рис. 1: Пример использования диалогового окна

13.1 jsAPI.modal.open(...)

`jsAPI.modal.open(options: IModalOptions, callbacks: IModalCallbacks)`

Аргумент	Тип	По умолчанию	Описание
`options.buttons`	`Array<IModalButton>`	–	Массив кнопок, отображаемых в нижней части диалогового окна

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
options.buttons[].class	string	–	CSS-класс для кнопки. Можно использовать для стилизации, к примеру: btn-danger
options.buttons[].text	string	–	Текст кнопки
options.buttons[].action	'accept' | 'close' | 'decline' | (btn?: IModalButton) => void	–	Действие при нажатии на кнопку. При передачи строки - будет выполнено соответствующее действие и отправлено системное событие. Если Вы определяете действие при нажатии самостоятельно (функцию), то система не будет применять никакие автоматические действия при ее нажатии.
options.centered	boolean	false	Вертикальное выравнивание по центру страницы
options.closeByEscape	boolean	true	Закрывать окно по нажатию Esc?
options.closeButton	boolean	true	Отображать кнопку закрытия окна?
options.minHeight	number	240	Минимальная высота окна в px
options.page	IPageRouteModal | string	–	URL - страницы, которая будет отображаться в диалоговом окне. Объект {src, query} или строка
options.page.src	string	–	номер страницы, которую необходимо отобразить
options.page.query	Record<string, string>	–	GET-параметры, которые следует передать при открытии страницы options.page.src в диалоговом окне

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.selector`	`string`	–	CSS-селектор элемента для привязки к динамическому действию (Dynamic Action, DA). События динамический действий с диалоговым окном будут вызваны для этого элемента `onModalClose`, `onModalDecline`, `onModalAccept`
`options.text`	`string`	–	Текст, отображаемый в контентной части диалогового окна
`options.title`	`string`	–	Заголовок окна
`options.width`	`number|string`	`null`	Ширина окна
`callbacks.onClose`	`(currentModal, items) => void`	–	Вызывается при закрытии окна с использованием методов `jsAPI.modal.close();`, `jsAPI.modal.accept();`
`callbacks.onAccept`	`(currentModal, items) => void`	–	Вызывается при закрытии окна с использованием метода `jsAPI.modal.accept();`
`callbacks.onDecline`	`(currentModal, items) => void`	–	Вызывается при закрытии окна с использованием метода `jsAPI.modal.close();`

Все callback-функции принимают два аргумента:

- `currentModal` - внутренний объект *IModal*, содержащий всю информацию о диалоговом окне;
- `items` - значения полей ввода из диалогового окна (если в нем есть поля ввода).

```
// Стандартный вызов страницы в модальном окне
jsAPI.modal.open({
    page: `/1000/`,
    title: "Заголовок"
});

// Вызов диалогового окна для подтверждения действия
jsAPI.modal.open(
    {
        title: "Заголовок",
        text: "Текст",
        width: 420,
```

(продолжается на следующей странице)

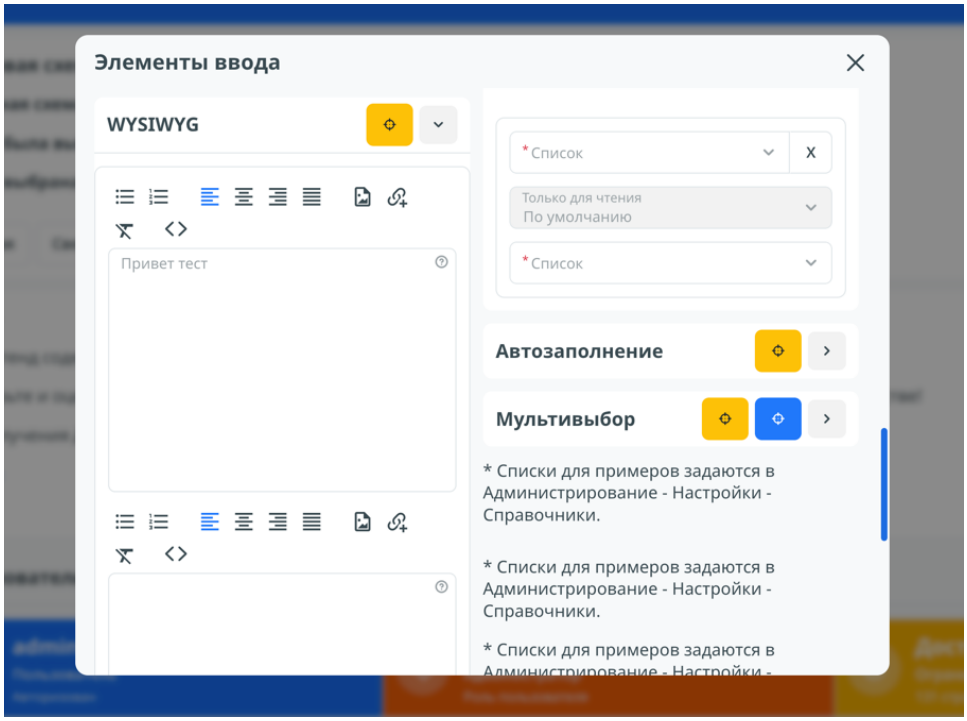

Рис. 2: Пример использования диалогового окна для вывода страницы

(продолжение с предыдущей страницы)

```
        buttons: [
            {
                text: "Да",
                action: "accept",
                class: "btn-primary"
            },
            {
                text: "Нет",
                action: "decline",
                class: "btn-secondary"
            },
            {
                text: "Я подумаю",
                action: function() {
                    console.log("Пользователь решил подумать");
                    jsAPI.modal.close();
                },
                class: "btn-danger"
            }
        ]
    },
    {
        onClose: function(currentModal, items) {
            // Выполнится при закрытии
        },
        onDecline: function(currentModal, items) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
            // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального␣
↪окна
        },
        onAccept: function(currentModal, items) {
            // Выполнится при нажатии кнопки "Да
        }
    }
);

// Открытие страницы в диалоговом окне с указанием GET параметров и селектором для␣
↪привязки к динамическому действию
jsAPI.modal.open({
    page: {
        src: 1000,
        query: {
            P2000_ITEM: jsAPI.getItem("P2000_ITEM"),
            clear: 1000
        }
    },
    title: "Заголовок",
    selector: "#DA_ELEMENT_ID" // У этого элемента будут вызваны события onModalClose␣
↪| onModalDecline | onModalAccept
}, {
    onClose: function(currentModal, items) {
        console.log('onClose', currentModal, items);
        // Выполнится при jsAPI.modal.close(), jsAPI.modal.accept()
    },
    onAccept: function(currentModal, items) {
        console.log('onAccept', currentModal, items);
        // Выполнится при jsAPI.modal.accept()
    },
    onDecline: function(currentModal, items) {
        console.log('onDecline', currentModal, items);
        // Выполнится при jsAPI.modal.close()
    }
});
```

13.2 jsAPI.modal.accept()

```
jsAPI.modal.accept()
```

Закрывает модальное окно и вызывает `onClose`, `onAccept`

13.3 jsAPI.modal.close()

```
jsAPI.modal.close()
```

Закрывает модальное окно и вызывает `onClose`, `onDecline`

13.4 jsAPI.modal.closeAll()

```
jsAPI.modal.closeAll()
```

Закрывает все модальные окна и вызывает у каждого `onClose`, `onDecline`

ГЛАВА 14

setLanguage(...)

`jsAPI.setLanguage(languageCode: string) : void`

Изменить язык отображения приложения.

Аргумент	Тип	По умолчанию	Описание
`languageCode`	`string`	–	Код языка

Таблица 2: Поддерживаемые языки

Код	Наименование
`ru`	русский
`en`	английский

```
// Установить язык системы - русский
jsAPI.setLanguage("ru");

// Установить язык системы - английский
jsAPI.setLanguage("en");
```

ГЛАВА 15

getLanguage(...)

`jsAPI.getLanguage() : string`

Получить текущий язык отображения приложения.

Возвращает - код языка.

Таблица 1: Поддерживаемые языки

Код	Наименование
`ru`	русский
`en`	английский

```
const lang = jsAPI.getLanguage();
console.log('В приложении активен язык: '+lang)
```

ГЛАВА 16

:notification

`jsAPI.notification`

Объект доступа к методам вывода уведомлений

16.1 jsAPI.notification.success(...)

`jsAPI.notification.success(message: string): void`

Уведомление об успешном действии

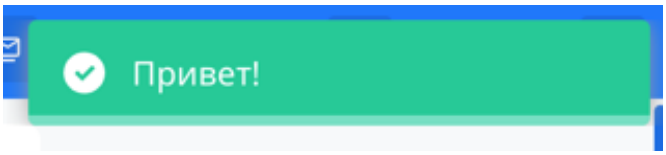

Рис. 1: Результат вызова jsAPI.notification.success

16.2 jsAPI.notification.error(...)

`jsAPI.notification.error(message: string): void`

Уведомление об ошибке

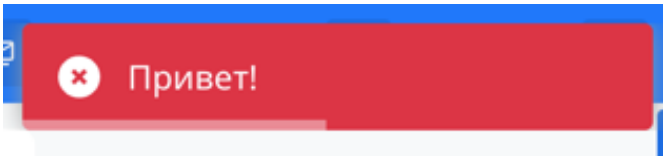

Рис. 2: Результат вызова jsAPI.notification.error

16.3 jsAPI.notification.warning(...)

```
jsAPI.notification.warning(message: string): void
```

Предупреждение

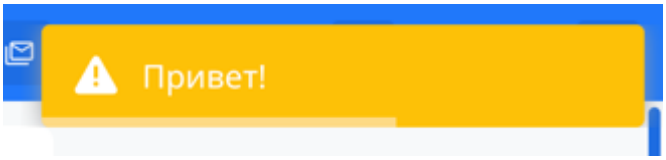

Рис. 3: Результат вызова jsAPI.notification.warning

16.4 jsAPI.notification.info(...)

```
jsAPI.notification.info(message: string): void
```

Информационное сообщение

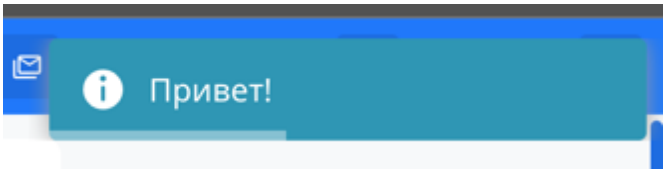

Рис. 4: Результат вызова jsAPI.notification.info

Аргумент **class** longtable	Тип	По умолчанию	Описание
`message`*	`stri` | `Arra`	–	одно сообщение если аргумент указан в виде строки, или массив сообщений если аргумент указан в виде массива строк (выводятся последовательно друг за другом)

```
// Сообщения об успешных действиях
jsAPI.notification.success(["Успешное действие", "Успешное действие 2"]);

// Сообщение об успешном действии
jsAPI.notification.success("Успешное действие");
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
// Сообщение об ошибке:
jsAPI.notification.error("Ошибка");

// Сообщения об ошибках
jsAPI.notification.error(["Ошибка", "Ошибка 2"]);

// Предупреждение (warning)
jsAPI.notification.warning("Внимание!"); //Выводит предупреждение (warning)

// Предупреждения (warnings)
jsAPI.notification.warning(["Предупреждение", "Предупреждение 2"]); //Выводит␣
↪предупреждения (warnings)

// Информационное сообщение (info)
jsAPI.notification.info("Внимание!"); //Выводит информационное сообщение (info)

// Информационные сообщения (info)
jsAPI.notification.info(["Предупреждение", "Предупреждение 2"]); //Выводит␣
↪информационные сообщения (info)
```

ГЛАВА 17

process(...)

```
jsAPI.process(requestName: string, items: Record<string, any>, callbacks?: IAppCallback)
: void
```

Вызов процесса, определенного ранее на стороне XRAD.

Метод *jsAPI.process* выполняет XHR-запрос `{API_URL}/callAction` и автоматически формирует параметры запроса.

Аргумент	Тип	По умолчанию	Описание
`requestName`	`string`	–	Имя запроса
`items`	`Record<string, any>`	–	Значения items, отправляемые в теле запроса `{API_URL}/callAction`. Допускается использовать пустой объект `{}` если не требуется отправки значений.
`callbacks.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`callbacks.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

```
// Вызов без дополнительных параметров
jsAPI.process("MY_PROCESS");

// Вызов с items
jsAPI.process("MY_PROCESS", {
    P1000_MY_ITEM_1: "Value 1",
    P1000_MY_ITEM_2: "Value 2"
});

// Вызов с items и обработчиком успешного ответа
jsAPI.process(
    "MY_PROCESS",
    {
        P1000_MY_ITEM_1: "Value 1",
        P1000_MY_ITEM_2: "Value 2"
        },
    {
        onSuccess: function(res) {
            console.log('onSuccess', res);
        }
    }
);

// Вызов с items, обработчиками успешного ответа и ошибки
jsAPI.process(
    "MY_PROCESS",
    {
        P1000_MY_ITEM_1: "Value 1",
        P1000_MY_ITEM_2: "Value 2"
    },
    {
        onSuccess: function(res) {
            console.log('onSuccess', res); // Ответ сервера
        },
        onError: function(err) {
            console.error('onError', err); // Ошибка сервера
        }
    }
);
```

ГЛАВА 18

redirect(...)

```
jsAPI.redirect(path: string, options?: IRedirectOptions) : void
```

Осуществляет переход на указанную страницу приложения (**внутренняя навигация**).

Аргумент	Тип	По умолчанию	Описание
`path`	`string`	–	Относительный путь до страницы
`options.behaviour`	`'push'|'rewrite'`	`push`	Тип перенаправления, который определяет поведение браузера при нажатии на кнопку „назад“. `push` - добавить к текущей истории браузера, `rewrite` - обновить последнюю посещенную страницу в истории браузера
`options.onSuccess`	`() => void`	–	Функция обратного вызова, которая будет вызвана после перехода на указанную страницу. Событие успешного перехода, будет вызвано даже если страница, на которую выполнено перенаправление отображается с ошибками.

Примечание Данный метод использует внутреннюю навигацию приложения и не является аналогом `window.location.href`. Изменение адреса страницы не предполагает ее перезагрузку.

```
// Стандартный вызов
jsAPI.redirect("/content/5000?clear=5000");

// Вызов с обработчиком
jsAPI.redirect("/content/5000?clear=5000", {
    onSuccess: function(data) {
        console.log(data);
    }
});
```

ГЛАВА 19

refreshList(...)

```
jsAPI.refreshList(data: Object, callbacks: IAppCallback) : void
```

Метод `jsAPI.refreshList` предназначен для обновления списков привязанных к элементам форм. Таких как Select List, Multiselect, Autocomplete.

Метод выполняет XHR-запрос `{API_URL}/callAction` и автоматически формирует параметры запроса.

Аргумент	Тип	По умолчанию	Описание
`data`	`Object`	–	Данные для отправки в теле запроса
`data.id`	`string`	–	Обязательный. Идентификатор региона
`data.items`	`Record<string, string>`	–	Значения элементов формы
`callbacks.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`callbacks.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Примечание: Для обновления элемента формы необходимо указать его `id` в параметре `data`.

```
// Стандартный вызов
jsAPI.refreshList({
    id: "P1000_SELECT",
    items: {
        P1000_PARAM: "New P1000_PARAM val"
    }
});

// Стандартный вызов с обработчиками
jsAPI.refreshList(
    {
        id: "P1000_SELECT",
        items: {
            P1000_PARAM: "New P1000_PARAM val"
        }
    },
    {
        onSuccess: function(data) {
            console.log('onSuccess', data);
        },
        onError: function(err) {
            console.error('onError', err);
        }
    }
);
```

ГЛАВА 20

reload(...)

`jsAPI.reload(): void`

Осуществляет перезагрузку страницы.

```
jsAPI.reload();
```

ГЛАВА 21

setLoading(...)

`jsAPI.setLoading(value: boolean) : void`

Устанавливает состояние загрузки для текущей страницы или модального окна

Аргумент	Тип	По умолчанию	Описание
`value`	`boolean`	–	**true** - отобразить состояние загрузки. **false** - скрыть состояние загрузки

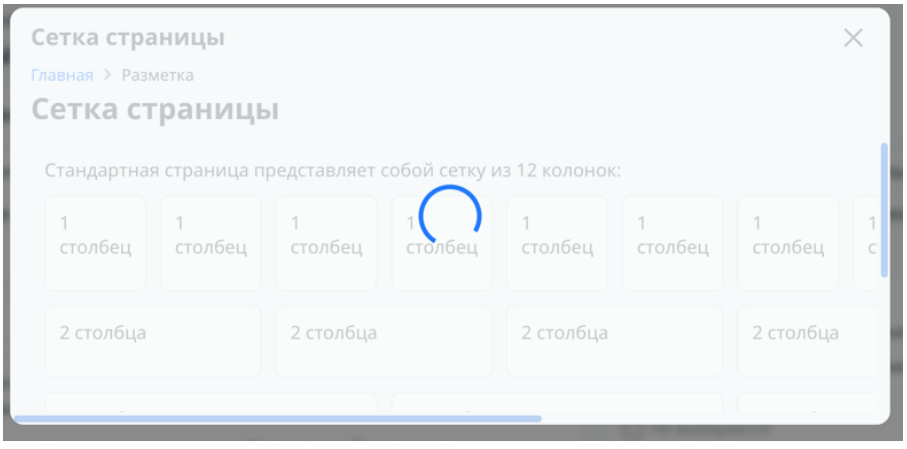

Рис. 1: Пример отображения состояния загрузки диалогового окна

```
jsAPI.setLoading(true);
jsAPI.setLoading(false);
```

ГЛАВА 22

submit(...)

```
jsAPI.submit(data: Record<string, any>, options?: ISubmitOptions): void
```

Осуществляет отправку данных страницы на сервер.

Аргумент	Тип	По умолчанию	Описание
`data`	`string | Object | string`	–	Объект с данными, которые Вы хотите отправить, либо строка, которая будет использована как параметр `request` в запросе
`data.items`	`Object`	–	Значения items, отправляемые в теле запроса `{APIURL}/processPage`, помимо основных items страницы. Допускается использовать пустой объект `{}`, если не требуется отправки кастомных значений.
`options.clientValidation`	`boolean`	`true`	Необходимо ли блокировать выполнение запроса, если валидация на стороне клиента выявила ошибки заполнения?
`options.setLoading`	`boolean`	`false`	Необходимо ли блокировать страницу и отображать „загрузку“ на время выполнения запроса?

продолжается на следующей странице

Таблица 1 – продолжение с предыдущей страницы

Аргумент	Тип	По умолчанию	Описание
`options.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`options.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Примечание: При указании обработчика `onSuccess` не будет запущена перезагрузка страницы (переопределение поведения по умолчанию).

Если требуется перезагрузка страницы и дополнительная логика в собственном обработчике, используйте метод `jsAPI.reload()` внутри `onSuccess`

```
// Стандартный вызов
jsAPI.submit({});

// Стандартный вызов с игнорированием ошибок валидации на стороне клиента
jsAPI.submit({}, { clientValidation: false });

// Стандартный вызов с автоматической блокировкой страницы на время выполнения запроса
jsAPI.submit({}, { setLoading: true });

// Вызов с дополнительными значениями items и обработчиком успешной отправки и ошибки
jsAPI.submit(
    {
        MY_CUSTOM_ITEM: "CUSTOM VALUE",
    },
    {
        onSuccess: function(res) {
            console.log("При кастомном обрабортчике не будет перезагрузки страницы");
        },
        onError: function(err) {
            console.error(err); //Обработчик ошибки сервиса {API_URL}/processPage
        }
    }
);

// Вызов без дополнительных значений items и с обработчиком успешной отправки
jsAPI.submit(
    {},
    {
        onSuccess: function(res) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
            //Здесь может быть любой код
            jsAPI.reload(); //И после него выполнится перезагрузка страницы
        }
    }
);
```

ГЛАВА 23

tooltip(...)

`jsAPI.tooltip(selector: string): TooltipActions`

Модуль для управления всплывающими подсказками.

Свойство	Тип	Описание
`selector`	`string`	CSS-селектор элемента, с которым планируется взаимодействовать

Возвращает набор методов для взаимодействия с всплывающими подсказками для элемента указанного в `selector`.

```
// Отобразить всплывающую подсказку для элемента "#TARGET" и скрывать при потере␣
↪фокуса
jsAPI.tooltip("#TARGET").show("Текст подсказки");

// Добавить всплывающую подсказку для элемента "#TARGET" и отобразить при наведении␣
↪на него, скрывать когда пользователь убрал курсор
jsAPI.tooltip("#TARGET").set("Текст подсказки");

// Скрыть всплывающую подсказку у элемента "#TARGET"
jsAPI.tooltip("#TARGET").hide();

// Удалить всплывающую подсказку у элемента "#TARGET"
jsAPI.tooltip("#TARGET").hide();
```

23.1 jsAPI.tooltip(...).set(...)

`jsAPI.tooltip(...).set(value: string) : void`

Добавить всплывающую подсказку.
Отображать ее **при наведении мыши**.
Когда пользователь **убрал мышь с элемента - скрывать**.

Свойство	Тип	По умолчанию	Описание
`value`	`string`	–	Текст всплывающей подсказки

23.2 jsAPI.tooltip(...).remove()

```
jsAPI.tooltip(...).remove() : void
```

Удалить всплывающую подсказку с элемента.

23.3 jsAPI.tooltip(...).show(...)

```
jsAPI.tooltip(...).show(value: string) : void
```

Добавить и **сразу отобразить** всплывающую подсказку.
Отображать ее **при наведении мыши**.
Скрывать ее **при нажатии вне области элемента**. (не скрывать когда пользователь лишь убрал мышь с элемента)

Свойство	Тип	Описание
`value`	`string`	Текст всплывающей подсказки

23.4 jsAPI.tooltip(...).hide(...)

```
jsAPI.tooltip(...).hide() : void
```

Скрыть всплывающую подсказку.
Скрыть, но не полностью удалить. Для полного удаления воспользуйтесь `jsAPI.tooltip(...).remove()`

23.5 jsAPI.tooltip(...).hideAll()

```
jsAPI.tooltip(...).hideAll() : void
```

Скрыть все всплывающие подсказки.
Скрыть, но не полностью удалить. Для полного удаления воспользуйтесь `jsAPI.tooltip(...).remove()`

ГЛАВА 24

:debug устарел

`jsAPI.debug`

Управление режимом отладки

Включение и отключение различных уровней отображения отладочной информации.
Изменения применяются только к текущему устройству пользователя.
При включенном режима отладки вы получите больше информации об ошибках выполнения сетевых запросов.

Таблица 1: Доступные уровни отладки `DebugType`

Имя	Описание
`CRITICAL`	Критические
`ERROR`	Ошибки системы
`WARNING`	Предупреждения
`INFO`	Информационные сообщения
`DEBUG`	Отладочная информация

24.1 debug.enable(...)

`jsAPI.debug.enable(type: DebugType): void`

Активировать режим отладки уровня `type`

Аргумент	Тип	По умолчанию	Описание
`type`	`DebugType`	–	Уровень отладки

```
//Активировать отладку с типом CRITICAL:*
jsAPI.debug.enable("CRITICAL");
```

24.2 debug.disable(...)

`jsAPI.debug.disable(type: DebugType): void`

Деактивировать режим отладки уровня `type`

Аргумент	Тип	По умолчанию	Описание
`type`	`DebugType`	–	Уровень отладки

```
// Деактивировать отладку с типом CRITICAL:
jsAPI.debug.disable("CRITICAL");
```

ГЛАВА 25

dom низкоуровневый

Методы для прямого взаимодействия с DOM-элементами.
Отличаются от нативных методов javascript тем, что имеют встроенные обработчики ошибок.
Предотвращают остановку скрипта в случае возникновении ошибки.

> *Мы не рекомендуем напрямую взаимодействовать с DOM-деревом. Старайтесь использовать методы компонентов там, где это возможно, чтобы избежать ошибок при обновлении системы. Изменения, примененные с использованием методов прямого доступа к DOM-дереву могут быть утеряны при выполнении отрисовки компонента (rerender).*

25.1 dom.blur()

```
jsAPI.dom.blur() : void
```

Удаляет фокус клавиатуры с текущего элемента.
Обёртка для нативного метода document.activeElement.blur()

25.2 dom.focus(...)

```
jsAPI.dom.focus(selector: string) : void
```

Устанавливает фокус на указанный элемент, если он может быть сфокусирован.
Обёртка для нативного метода el.focus()

Аргумент	Тип	По умолчанию	Описание
selector	string	–	CSS-селектор элемента

25.3 dom.hide(...)

`jsAPI.dom.hide(selector: string) : void`

Скрывает элемент, добавляя к нему системный CSS-класс `js-api-hidden`

Аргумент	Тип	По умолчанию	Описание
selector	string	–	CSS-селектор элемента

25.4 dom.setAttribute(...)

`jsAPI.dom.setAttribute(selector: string, attrs: Record<string, number|string>) : void`

Добавляет и изменяет HTML-атрибуты у элемента

Аргумент	Тип	По умолчанию	Описание
selector	string	–	CSS-селектор элемента
attrs	Record<string, number|string>	–	Атрибуты для установки: атрибут-значение

```
jsAPI.dom.setAttribute("#P1000_MY_ITEM", {
   disabled: true,
   "data-custom-attr": "My custom value"
 });
```

25.5 dom.show(...)

`jsAPI.dom.show(selector: string) : void`

Отображает элемент с указанным css селектором путём удаления css класса `js-api-hidden`

Аргумент	Тип	По умолчанию	Описание
selector	string	–	CSS-селектор элемента

ГЛАВА 26

changeTab(...) устарел

```
jsAPI.changeTab(id: string, tab: number) : void
```

Устарел. Используйте jsAPI.component(…).changeTab(…)

Переключает активную вкладку для региона с типом «Tabs».

Используйте метод *changeTab* для компонента *Tab*

Аргумент	Тип	По умолчанию	Описание
`id`	`string`	–	id региона с типом „Tabs“
`tab`	`number`	–	Порядковый номер вкладки (начиная с 1)

```
jsAPI.changeTab("MY_TABS", 2);
```

ГЛАВА 27

clearLocalStorage() низкоуровневый

```
jsAPI.clearLocalStorage() : void
```

Удаляет все записи в localStorage приложения.
Является обёрткой нативного метода window.localStorage.clear();

> *При очищении хранилища вы можете столкнуться с потерей данных, хранящихся на стороне устройства пользователя. Убедитесь, что вы осознаете все риски, связанные с выполнением операции очистки перед применением этого метода в вашем приложении.*

```
jsAPI.clearLocalStorage();
```

ГЛАВА 28

confirmModal(...) устарел

```
jsAPI.confirmModal(options: IConfirmOptions, callbacks: ICallbacks) : void
```

Устарел. Используйте modal/open

Отображает окно для подтверждения какого-либо действия.

Данный метод является частным случаем вызова modal/open

```
jsAPI.modal.open(
   {
     title: "Заголовок",
     text: "Текст",
     width: 420,
     buttons: [ {
         text: "Да",
         action: "accept",
         class: "btn-primary"
       },
       {
         text: "Нет",
         action: "decline",
         class: "btn-secondary"
       }
     ]
   },
   {
     onClose: function(e) {
       // Выполнится при закрытии
     },
     onDecline: function(e) {
       // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального окна
     },
     onAccept: function(e) {
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
      // Выполнится при нажатии кнопки "Да
    }
  }
);
```

Таблица 1: IConfirmOptions

Аргумент	Тип	По умолчанию	Описание
`options.title`	`string`	–	Заголовок окна
`options.text`	`string`	–	Текст, отображаемый в диалоговом окне

Таблица 2: ICallbacks

Аргумент	Тип	По умолчанию	Описание
`callbacks.onAccept`	`() => void`	–	Выполняется при нажатии кнопки „Да“
`callbacks.onDecline`	`() => void`	–	Выполняется при нажатии кнопки „Нет“ и стандартном закрытии
`callbacks.onClose`	`() => void`	–	Выполняется при закрытии окна

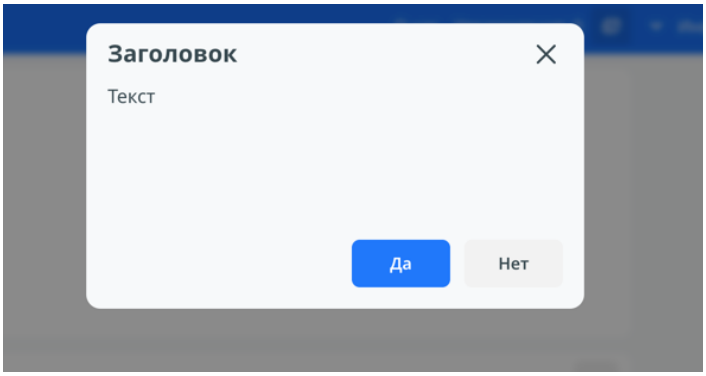

Рис. 1: Пример выполнения функции ниже

```
// Еще один пример использования метода jsAPI.confirmModal
jsAPI.confirmModal(
  {
    title: "Вы уверены?",
    text: "При закрытии данные не сохранятся"
  },
  {
    onDecline: function(e) {
      // Выполнится при нажатии кнопки "Нет" и стандартном закрытии модального окна
    },
    onAccept: function(e) {
      // Выполнится при нажатии кнопки "Да
```

(продолжается на следующей странице)

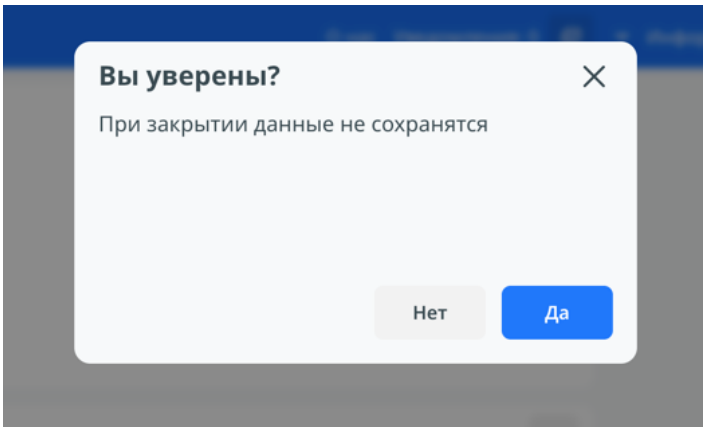

Рис. 2: Пример выполнения функции ниже

(продолжение с предыдущей страницы)

```
      }
    }
);
```

ГЛАВА 29

refresh(...) устарел

`jsAPI.refresh(data: Object, callbacks: IAppCallback)`

Устарел. Используйте *jsAPI.component(…).refresh(…)*

Метод *jsAPI.refresh* выполняет XHR-запрос `{API_URL}/callAction` и автоматически формирует параметры запроса.

Аргумент	Тип	По умолчанию	Описание
`data`	`Object`	–	Данные для отправки в теле запроса
`callbacks.onSuccess`	`string`	`(res: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **успешно**. Принимает один аргумент - ответ сервера (res)
`callbacks.onError`	`string`	`(err: unknown) => void`	Функция обратного вызова, которая сработает, если запрос завершился **с ошибкой**. Принимает один аргумент - возникшая ошибка/ошибки (err)

Примечание: Для обновления региона необходимо указать его `id` в параметре `data`:

```
// Стандартный вызов
jsAPI.refresh({
    id: "PARAMS",
    items: {
        P1000_ID: "1"
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    }
});

// Стандартный вызов с обработчиками
jsAPI.refresh(
    {
        id: "PARAMS",
        items: {
            P1000_ID: "1"
        }
    },
    {
        onSuccess: function(data) {
            console.log('onSuccess',  data);
        },
        onError: function(err) {
            console.error('onError', err);
        }
    }
);
```

ГЛАВА 30

updateItem(...) устарел

`jsAPI.updateItem(id: RegionId, props: Record<string, any>): void`

Устарел. Используйте *jsAPI.component(…).properties(…)*

Обновляет параметры элемента формы на уровне страницы.

Аргумент	Тип	По умолчанию	Описание
`id`	`RegionId`	–	id элемента формы. НЕ является селектором (указывать без '#')
`props`	`Object`	–	Новые значения для свойств компонентов

Таблица 2: Поддерживаемые свойства

Имя	Тип	Описание
`label`	`string`	Подпись для элемента формы
`required`	`boolean`	Обязательность заполнения
`col`	`number`	Ширина колонки в форме (от 1 до 12)
`mask`	`string`	Маска поля (работает только с TEXT)
`maxlength`	`number`	Макс. кол-во символов (работает только с TEXT)

```
jsAPI.updateItem("P1_LOGIN", {
    label: "Новый лейбл",
    required: false,
    col: 6,
    mask: "999",
```

(продолжается на следующей странице)

(продолжение с предыдущей страницы)

```
    maxlength: 3
});
```