Рекомендации по приведению сайта к требованиям WCAG 2.0 AA

Введение

Стандартом WCAG 2.0 определяются требования по доступности сайта, внутри стандарта существует три уровня соответствия: A — низший, AA — средний и AAA — наивысший. В этом документе рассмотрен уровень AA.

Стандарт довольно подробно описывает все требования, содержит ссылки на разъяснения, используемые техники и часто встречающиеся ошибки. Это позволяет полностью опираться на него при анализе доступности веб-страниц и их адаптации. Кроме того, в общем доступе имеется большой инструментарий для полуавтоматического выявления ошибок: валидаторы и дополнения для браузеров.

В этом документе представлены практические рекомендации по применению стандарта, описан инструментарий и процесс проверки сайта на доступность, даны ссылки на ключевые материалы и руководства по доступности сайтов, а также на примеры реализации наиболее распространённых виджетов.

Стандарт WCAG 2.0 уровня AA описывает критерии доступности для разных категорий пользователей. На практике для соблюдения хорошего уровня соответствия критериям тестировщику будет полезно представить себя на месте следующих категорий пользователей (или привлечь их к тестированию):

- Страдающие различными видами дальтонизма (цветовой слепоты). Там, где цвет используется для индикации или предоставления информации, должны быть предусмотрены альтернативные визуальные средства.

- Слабовидящие. Накладываются требования по контрастности, размеру элементов и поддержке масштабирования страницы.

- Слепые (полностью незрячие - пользователи экранных чтецов). Требуется широкий набор мер, включая:

    - предоставление текстовой альтернативы для всех значимых нетекстовых элементов,
    - предоставление семантической вёрстки,
    - правильное использование семантических областей, заголовков и других навигационных элементов,
    - предоставление дополнительных средств навигации по странице,
    - предоставление дополнительной метаинформации об элементах на странице и связях между ними,

•обязательное предоставление текстовых меток и, при необходимости, подсказок для элементов ввода,
•учет особенностей восприятия содержимого: оно воспринимается на слух, т.е. последовательно, без возможности охвата всей страницы взглядом целиком, без возможности заметить информацию в другой области страницы (если она не связана соответствующим образом), без восприятия сенсорных характеристик текста,
•реализация взаимодействия с элементами управления или ввода с помощью набора атрибутов (role, aria-* и др.).

•Пользователи с нарушением слуха. Требуется обязательное предоставление текстовой альтернативы для аудиоконтента.

•Пользователи с нарушением опорно-двигательного аппарата, которое может выражаться в неспособности пользоваться мышью. Требуется полная управляемость и доступность сайта для клавиатуры.

Всем участникам разработки (дизайнерам, разработчикам, проект-менеджерам, тестировщикам и редакторам) можно также ознакомиться с кратким чек-листом, в котором описаны основные обязанности каждого специалиста по обеспечению доступности.

См. также:

•Раздел "Введение" стандарта WCAG 2.0

•Методики выполнения WCAG 2.0

Дизайн

1.Цвет не может быть единственным способом предоставления какой-либо информации или разметки.

2.Необходимо соблюдать контрастность согласно WCAG 2.0.

Измерять контрастность в HTML можно WAVE Evaluation Tool

3.Не следует опираться на сенсорные характеристики.

Детально данное требование разобрано ниже в описании критерия 1.3.3.

4.Должно быть предусмотрено визуальное отображение состояний фокуса на полях и элементах управления.

5.Для информации, которую принципиально нельзя сделать доступной, следует предусматривать альтернативные формы представления или, если даже это невозможно, предоставить текстовое описание возможностей интерфейса (например, чтобы слепой пользователь имел возможность понять функционал страницы и обратиться за помощью к зрячему).

См. также Accessibility Guidelines. The Checklist for Designers.

Средства проверки

1.Проверка сайта валидаторами на общую валидность
HTML: https://validator.w3.org/

2.Проверка сайта специализированными валидаторами:

•http://achecker.ca/checker/ валидатор на основе анализа HTML-кода. Не учитывает JavaScript, ошибки выводятся в текстовом виде, не очень удобны для анализа. Выдаёт список потенциальных проблем, большая часть из которых не соответствуют действительности, однако список может использоваться в качестве дополнительного чек-листа.

•http://wave.webaim.org/ и расширение для Chrome WAVE Evaluation Tool. Находят большое количество ошибок, предоставляют подсказки по исправлению и ссылку на стандарт, предоставляют информацию в удобном виде, подсвечивают синтаксис и aria-атрибуты.

Главное удобство в том, что расширение может анализировать не исходный код, а текущее состояние страницы.

•http://khan.github.io/tota11y/ - встраиваемый на страницу с помощью букмарклета визуализатор и валидатор доступности. Поддерживает измерение контрастности. В отличие от Wave Evaluation Tool, не показывает все ошибки сразу и определяет не так много ошибок.

•AInspector Sidebar for Firefox. В отличие от предыдущих валидаторов, позволяет находить ошибки в логике применения aria-атрибутов. Кроме списка ошибок выдает также список проверок, которые рекомендуется сделать вручную.

•Accessibility Developer Tools Chrome. Отлавливает в основном формальные ошибки применения aria-атрибутов. Из минусов: неудобный интерфейс и непонятный формат вывода ошибок.

Нужно помнить, что наличие сообщений об ошибках валидатора само по себе не является формальным критерием несоответствия WCAG. Кроме того, некоторые сообщения носят рекомендательный характер или помечены как потенциальные. Решения по ним должны приниматься исходя из контекста.

Верно и обратное: не любая проблема находится валидатором: необходимо ручное тестирование в т.ч. с применением экранных чтецов (см. ниже).

3.Проверка в экранных чтецах.

Наиболее простой в настройке является комбинация Windows + Firefox/IE + NVDA. Наиболее широко распространён JAWS (программа платная и дорогая, поставляется в триал-версии с 40-минутным режимом, но лицензия JAWS запрещает его использование в демонстрационном режиме для целей тестирования).

Пользователям других ОС тестовое окружение можно настроить в виртуальных машинах от Microsoft (https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/ - бывший modern.ie), работает приемлемо, по крайней мере, с ВМ Windows 7.

Программы экранного доступа довольно специфичны для тех, кто с ними сталкивается впервые, однако к пользованию ими можно относительно быстро привыкнуть. Освоить программу на уровне краткой инструкции (см. ниже) рекомендуется всем фронт-энд разработчикам и тестировщикам. Это не займёт много времени.

•Важно! Описание принципов работы программ экранного доступа: http://webaim.org/techniques/screenreader/
•NVDA: бесплатная программа экранного доступа для тотально незрячих и слабовидящих:http://www.nvaccess.org/.
•Краткая инструкция по пользованию NVDA для зрячих разработчиков: http://webaim.org/articles/nvda/
•Шпаргалка по командам NVDA: http://webaim.org/resources/shortcuts/nvda. (NVDA key - это Insert и/или Caps Lock в зависимости от настроек).
•Развернутый список команд NVDA: https://dequeuniversity.com/screenreaders/nvda-keyboard-shortcuts.

•Программа экранного доступа
JAWS: http://www.freedomscientific.com/Products/Blindness/JAWS.
•Шпаргалка по командам
JAWS: http://webaim.org/resources/shortcuts/jaws.
•Таблица поддержки различных техник в разных сочетаниях браузеров и экранных чтецов: http://www.powermapper.com/tests/. Содержит также информацию о восприятии невалидного с точки зрения WCAG кода.

Изначально при вёрстке нужно учитывать

1.Соблюдение семантики разметки. В частности, можно выделить следующее (перечислено далеко не всё):

•Разметка списков, перечислений пунктов, лент документов и пр. тегами <ul>, <ol>, <li> или соответствующими атрибутами role.
•Разметка табличных данных тегами <table>, <tr>, <th>, <td> или соответствующими атрибутами role. Обратите внимание Правила вложенности элементов для атрибутов role полностью аналогичны правилам для тегов.
•Использование тегов <caption> и <summary>
•Правильное использование заголовков таблицы. Нужно учесть, что заголовками ячейки являются все вышестоящие элементы <th> во всей таблице. Заголовки с colspan применяются к каждой нижестоящей ячейке во всех затронутых столбцах.
•Не разбивать таблицу или список на несколько только для отображения.
•Кнопки рекомендуется оформлять с помощью кликабельных элементов: <button>, <a href... role="button">, <input type="button">. Допускается также использование некликабельных элементов с role=button, при условии фокусируемости (tabindex) и обработки событий клавиатуры (keydown на нажатие enter).

2.Разметка семантических областей с помощью role=main, role=navigation, role=contentinfo, role=complementary, role=banner и др.

> •Любой контент страницы должен принадлежать какой-либо семантической зоне.

> •Если на одной странице зоны с role равным navigation или complementary встречаются более одного раза, то им следует добавлять текстовые подписи, поясняющие их назначение посредством атрибута aria-label.

> •Добавление ссылки для пропуска повторяющихся блоков и перехода к блоку role="main".
>
> Ссылка должна быть первым фокусируемым элементом на странице. Это ссылка, предназначенная для незрячих пользователей. После загрузки страницы фокус должен попадать на ссылку по первому нажатию TAB, затем по нажатию на ENTER страница должна якориться на элементе с основным содержимым.
>
> Эталонная реализация на http://webaim.org/. После загрузки нужно нажать на TAB - ссылка становится видимой в левом верхнем углу, затем ENTER.

3.Строгая иерархия заголовков начиная с заголовка уровня 1.

4.Управляемость с клавиатуры:

•Все элементы управления и ввода должны быть фокусируемы
•Состояние фокуса должно быть различимо

5.Предоставление дополнительной текстовой информации:

•Атрибут alt: пустой у декоративных элементов и осмысленный текст для информативных элементов.

•Предоставление меток (label) для элементов ввода: с помощью <label for="...">, aria-label, aria-labelledby. Не должно быть элементов управления и ввода без текста или текстовой метки.

Подробнее о том, как вычисляется текстовая альтернатива, см. https://www.w3.org/TR/wai-aria/roles#textalternativecomputation

Обратите внимание, что не имеет смысла добавлять aria-label к не имеющим семантики элементам (например,).

Для прочих элементов (списки, группировки полей <fieldset> и др., landmarks) атрибут aria-label также будет интерпретироваться по-разному, и при его использовании нужно понимать его предназначение для каждого элемента.

•Отображение ошибок: общее сообщение об ошибке (в заголовке страницы - title, либо в начале блока основного содержимого страницы, либо в элементах с role="alert" или aria-live="assertive".

•Использование тегов <caption> и <summary> для описания таблиц, предназначение которых непонятно из предыдущего контекста или навигация по которым может потребовать дополнительных сведений.

•Раскрытие сокращений с помощью <abbr title> (вместо содержимого тега зачитывается значение title):

```
<abbr title="и так далее">и т.д.</abbr>
```

•Для элементов, смысл которых становится понятен только с учётом положения на странице или внешнего вида, начертания шрифта, зачеркнутого текста, представленной с помощью иконок информации (например, звёздность отеля), требуется текстовое описание, возможно, скрытое с помощью выноса за левый край экрана:

```
.sr_only {
  position: absolute;
  width: 1px;
  height: 1px;
  padding: 0;
  margin: -1px;
  overflow: hidden;
  clip: rect(0, 0, 0, 0);
  border: 0;
}
<span class="sr_only">текст для экранных чтецов</span>
```

•Значимая информация, представленная в виде диаграмм, графиков, интерактивных Flash (в большинстве случаев), SVG, Canvas и других, должна быть представлена также и в текстовом виде: отдельными параграфами, таблицами, возможно, на отдельной странице или в скрытом от зрячих пользователей с помощью class="sr_only" блоке.

Примечание: в большинстве случаев имеется техническая возможность адаптировать перечисленные элементы для пользования незрячими, но в виду трудозатратности, необходимости проработки по сути отдельного интерфейса для незрячих, проблем с тестированием и т.д. обычно легче и, главное, удобнее для пользователя иметь текстовое представление данных.

6.Рекомендуется по возможности использовать встроенные компоненты браузера, если они удовлетворяют требуемому функционалу: комбинированные списки (<select>), флаги (checkbox), радиокнопки, кнопки, поля ввода.

Следует также избегать переусложнения управления с клавиатуры, не изобретать новые паттерны взаимодействия при наличии решающих те же задачи стандартных решений, стараясь компоновать интерфейс из известных и доступных для инвалидов компонентов и подходов. См. раздел "Примеры реализации доступных интерфейсов" и описание критерия 4.1.2 ниже.

7.Допускается создание элементов, скрытых для зрячих пользователей, но доступных для незрячих. Делается это с помощью техники вынесения элемента далеко за левый край экрана (класс sr_only, описанный выше).

8.Допустима реализация отдельного альтернативного интерфейса для экранных чтецов (со скрытием основного варианта интерфейса) в случае, если обычный виджет сделать доступным затруднительно. При этом нужно не забывать об управляемости элемента с клавиатуры зрячими пользователями.

Пользоваться этим приёмом нужно с осторожностью: если есть выбор, то следует предпочесть адаптировать общий интерфейс под использование незрячими пользователями.

Реализация отдельного интерфейса смысл, если это значительно упрощает интерфейс для незрячих, например, ввод даты вручную вместо календаря (но календарь тоже можно сделать доступным), использование списка вместо карты для выбора стран/городов.

Важно не переусердствовать: практика показывает, что слепые пользователи не испытывают затруднения при пользовании некоторыми интерфейсами, которые на первый взгляд кажутся неудобными для незрячих.

Быстрая проверка

1.Проверка сайта валидаторами на общую валидность HTML: https://validator.w3.org/.

Максимальное соответствие спецификации используемой версии HTML не является обязательным, но рекомендуется (см. технику G192). Однако имеется ряд ошибок, важных для доступности сайта и обязательных к исправлению. Они перечислены в описании критерия 4.1.2 в разделе "Детальная проверка на соответствие WCAG AA".

2.Проверка сайта специализированными валидаторами. См. раздел "Средства проверки".

3.Проверка управляемости с клавиатуры без экранных чтецов.

Не должно быть кликабельных, но недоступных с клавиатуры элементов (если им нет специальной доступной альтернативы). Такие элементы следует реализовывать с помощью тегов , <button> или с помощью сочетания атрибутов role=button, role=link и tabindex.

Фокус должен быть видимым, корректно перемещаться, не "застревать" при попадании ни на один элемент и не теряться при любом действии пользователя в любом состоянии страницы.

4.Просмотр сайта с применёнными стилями, приближающими его к тому, каким его видят незрячие. Стили можно найти по адресу https://github.com/Harut/wai-aria.css. Это позволит найти большую часть ошибок «на глаз», не сверяясь с каждым пунктом чек-листа. Этот пункт является необязательным, не заменяет, а предваряет просмотр страницы в экранных чтецах. Обращать внимание, в первую очередь, рекомендуется на несоответствия в полной визуальной версии и версии с применёнными стилями.

5.Проверка в экранных чтецах. На этом этапе большинство критических ошибок должно быть обнаружено и исправлено при предыдущих проверках.

Необходимо проверить восприятие экранными чтецами таблиц, нестандартных элементов, удобство пользования функционалом страницы, правильность и полноту озвучиваемых атрибутов (в основном, role и aria-*.

6.Проверка форм в экранных чтецах требует особого внимания. Нужно проверить корректность всех текстовых меток, ошибок и инструкций, проверить поведение формы при успешной отправке и наличии ошибок, последовательность и полноту предоставления информации в режиме

заполнения формы (при переключении между полями с помощью TAB, а не в режиме чтения страницы), корректное перемещение фокуса и т.д.

Примеры реализации доступных интерфейсов

•Техники WCAG (ими удобнее пользоваться, переходя со ссылок в стандарте WCAG 2.0): https://www.w3.org/TR/WCAG20-TECHS/.

•Отдельно можно ознакомиться с ARIA Techniques for WCAG 2.0: https://www.w3.org/TR/WCAG20-TECHS/aria.html.

•WAI-ARIA 1.0 Authoring Practices: https://www.w3.org/TR/wai-aria-practices/. Документация для разработчиков, содержит описание неочевидных моментов и подходов по созданию доступного интерфейса.

Содержит также набор шаблонов для проектирования самых распространённых виджетов (эту информацию в более сжатом табличном виде можно также найти в рекомендациях от WebAIM).

•Open Ajax Accessibility Examples: http://oaa-accessibility.org/. Большой набор образцов реализации доступных компонентов. Стоит воспринимать не как готовые виджеты, а как примеры реализации.

•Руководство по доступности и ARIA от Mozilla: https://developer.mozilla.org/en-US/docs/Web/Accessibility. Также содержит набор готовых компонентов (некоторые из них, правда, не открываются):https://developer.mozilla.org/en-US/docs/Web/Accessibility/ARIA/widgets/overview

•Небольшой набор хорошо проработанных виджетов: http://heydonworks.com/practical_aria_examples/.

Детальная проверка на соответствие WCAG AA

В данном разделе приведена выжимка из WCAG в форме чек-листа на соответствие уровню AA, сгруппированный по соответствующим критериям WCAG. Список составлен на основе http://webaim.org/standards/wcag/checklist. Официальный набор рекомендаций, техник и список часто встречающихся ошибок можно найти по ссылке: https://www.w3.org/WAI/WCAG20/quickref/.

Всем участникам разработки и тестирования сайта рекомендуется внимательно изучить и освоить данный список, чтобы допускать как можно меньше ошибок на этапе разработки или исправлять их на ранних этапах. Это позволит обходиться наименьшими трудозатратами при разработке и тестировании, а также сосредоточиться при тестировании на деталях, которые важны, но могут быть упущены в общем количестве ошибок.

1.1. Текстовая версия: предоставьте текстовую версию любого нетекстового контента

1.1.1. Нетекстовый контент (Level A)

•Все изображения, кнопки изображения (form image buttons), и области image map имеют соответствующий эквивалентный альтернативный текст.

•Изображения, которые не представляют какого-либо содержимого, являются декоративными или содержание которых уже представлено текстом, имеют пустой атрибут alt="" или выполнены в виде фоновых изображений CSS.

•Все изображения-ссылки имеют альтернативный текст (aria-label у ссылки или alt у изображения).

•Для сложных изображений на той же или отдельной странице имеется развернутый текстовый аналог. Изображение может быть связано с текстом с помощью ссылки или атрибута longdesc.

•Все кнопки имеют осмысленный текст.

•Все поля ввода имеют осмысленные текстовые метки.

•Встроенные медиа-объекты должны быть озаглавлены или иным образом идентифицированы текстом, доступным для программ экранного доступа.

•Встроенные фреймы имеют осмысленные названия.

1.2. Медиаконтент: предоставьте альтернативную версию медиаконтента, ограниченного по времени

Вкратце, следует предоставлять расшифровку текста записи, текстовое описание её содержимого или субтитры. Если видео содержит визуальную информацию, которая не представлена звуком, следует также предоставить для него аудио-описание.

В случае, если такое содержимое появится на сайте, следует обратиться к стандарту WCAG или, например, рекомендациям Webaim.

1.3. Адаптируемость: создавайте контент, который можно представить в различных видах без потери данных или структуры

1.3.1. Информация и связность (Level A)

•Семантически значимые элементы использованы по предназначению в соответствии со спецификацией HTML.

•Для обозначения заголовков (<h1>), списков (<ul>, <ol>, and <dl>), специального или выделенного текста (например, , <code>, <abbr>, <blockquote>) и прочих значимых элементов использована семантическая вёрстка.

•Для табличных данных используются таблицы. Ячейки корректно связаны со своими заголовками по горизонтали и/или вертикали. Если содержимое таблицы требует пояснения, использованы теги <caption> и <summary>.

•Связанные поля формы сгруппированы в <fieldset>, содержащий осмысленный <legend>.

1.3.2. Значимая последовательность чтения (Level A)

•Порядок чтения и навигации по странице, определяемый порядком элементов в HTML-коде, интуитивен и логически обоснован, не искажает сути содержимого.

1.3.3. Сенсорные характеристики (Level A)

Если это не является неотъемлемой и неминуемой частью функционала:

•В тексте отсутствуют отсылки к форме, размерам и расположению элементов, указания по пользованию страницей не завязаны на этих характеристиках (например, "Нажмите на квадратную иконку", "Инструкцию можно найти в правой колонке" и т.д.).

•Использование страницы не завязано на звуке (например, "Продолжите после звукового сигнала").

1.4. Избирательность: упростите просмотр и прослушивание контента, отделив важные части от второстепенных

1.4.1. Использование цвета (Level A)

•Цвет не используется в качестве единственного средства предоставления контента, индикации или различия элементов.

•Цвет не используется как единственное средство обозначения ссылок на фоне остального текста, за исключением случая, когда контраст по яркости между цветами текста и ссылок не менее 3:1, и при навигации или фокусе ссылка получает дополнительные различия (например, подчеркивание).

1.4.2. Управление звуком (Level A)

•Если на странице присутствует звук, который автоматически проигрывается более 3 секунд, необходимо дать возможность его остановить, поставить на паузу, заглушить или настроить его громкость.

1.4.3. Контраст (Level AA)

•Текст и текст на изображениях должны иметь коэффициент контрастности не менее 4,5:1.

•Увеличенный текст и изображение увеличенного текста имеют коэффициент контрастности не менее 3:1.

1.4.4. Изменение размеров текста (Level AA)

•Желательно, чтобы страница оставалась читаемой и функциональной при увеличении масштаба в пределах до 200%. Особенно это актуально для блоков, содержащих мелкий и низкоконтрастный текст.

1.4.5. Текст на изображениях (Level AA)

•Если можно добиться такого же визуального представления посредством доступного текста, и если содержание текста в изображении не имеет ключевого значения (логотип, скриншот и т.п.), то не следует использовать изображения, содержащие текст.

2.1. Доступность управления с клавиатуры

2.1.1. Клавиатура (Level A)

•Весь функционал должен быть доступен для управления с клавиатуры, за исключением случаев, когда это в принципе невозможно (например, рисование от руки).

Необходимо убедиться в работоспособности как в сочетании с экранными чтецами, так и без.

•Все элементы, с которыми можно взаимодействовать, должны принимать фокус. В случае, если реализовано нажатие на элемент, оно должно быть доступно наравне как посредством мыши, так и посредством клавиатуры.

По умолчанию, фокусировку и взаимодействие с клавиатурой поддерживают элементы формы, кнопки <button> и ссылки <a href>.

•Собственные реализации элементов управления или ввода должны предоставлять те же возможности для взаимодействия посредством клавиатуры, что и встроенные в браузер элементы и/или примеры реализации похожих виджетов на специализированных сайтах по доступности (например, Open Ajax Accessibility).

Нужно помнить, что реализации виджетов на этих сайтах может быть не идеальной, и во многих случаях может потребоваться их доработка или исправление. Необходимо проверять сложные решения в экранных чтецах.

•Рекомендуется избегать использования атрибута accesskey без необходимости. Атрибут accesskey нужно применять с осторожностью: он не должен конфликтовать с популярными горячими клавишами экранных чтецов (JAWS, NVDA). На русской версии рекомендуется выбирать accesskey таким, чтобы символ располагался на одной и той же кнопке в русской и английской раскладке.

•Задание положительного tabindex может вызвать проблемы с порядком переключения фокуса.

•Использование обработчиков mousedown и mouseup в качестве обработчиков нажатия на элемент будет приводить к недоступности его с клавиатуры. Также не обеспечивает доступность событие click на не фокусируемых по-умолчанию элементах, т.е. всех, кроме элементов формы, кнопок <button> и ссылок <a href>.

Рекомендуется избегать таких случаев, но если это невозможно, то для таких элементов необходимо дополнительно прописывать обработку событий клавиатуры.

•Открывающиеся при наведении мыши подсказки или меню также должны быть доступны с клавиатуры. Можно, например, показывать (а для подсказок ещё и озвучивать) текст элемента при фокусе.

Спецификация ARIA предусматривает атрибут aria-haspopup для реализации подобных виджетов (например), но с его поддержкой и работой с клавиатуры без экранных чтецов есть вопросы.

•Экранные чтецы могут переопределять обработку нажатия некоторых клавиш (например, стрелок клавиатуры или букв в режиме чтения - без фокуса на элементах формы), поэтому к обработке нажатия клавиш вне полей формы нужно относиться с осторожностью, изначально продумывать и впоследствии тестировать их поведение в сочетании с экранными чтецами; переопределение обработки нажатия TAB также может вызвать проблемы с порядком переключения фокуса.

2.1.2. Отсутствие ловушек для фокуса (Level A)

•Фокус при попадании на любой элемент или группу элементов никогда не застревает в них, и может быть свободно перемещён на любой доступный для управления элемент на странице.

2.2. Достаточное время: предоставьте пользователям достаточно времени для ознакомления и работы с контентом

2.2.1. Настройка времени (Level A)

•Если страница или приложения имеют временные ограничения, то у пользователя должна быть возможность его выключить, настроить, продлить.

•Исключения, предусмотренные стандартом:

•работающие в реальном времени приложения;
•случаи, когда ограничения по времени имеют ключевое значение и не могут быть убраны без ущерба для функционала, информационной значимости страницы или безопасности;
•ограничения по времени более 20 часов.

2.2.2. Пауза, остановка, скрытие. (Level A)

•Пользователь должен иметь возможность остановить, поставить на паузу или скрыть длящиеся более 5 секунд движения, мерцания или перемотку любых элементов. Это не относится к индикаторам загрузки.

•Для анимации переходов и привлечения внимания пользователей можно использовать движение, мерцание или перемотку длительностью менее 5 секунд.

•Также должна иметься возможность поставить на паузу автоматическое обновление любого содержания, за исключением случаев, когда обновление имеет ключевое значение, и без него страница теряет смысл.

Стандарт предусматривает исключения из правил, если отсутствует техническая возможность или если анимация или обновление имеют ключевое значение для функционала страницы, и без них меняется смысл или поведение страницы. Примеры исключений с объяснением можно найти в пояснениях к критерию.

Критерий довольно строгий, и полное следование ему иногда может потребовать много времени на разработку и ухудшить пользование страницей для обычных пользователей. Нужно искать компромисс между строгими формальными требованиями стандарта и реальностью в каждом случае, но при этом нужно продумывать взаимодействие для каждой из перечисленных категорий пользователей (см. Введение).

2.3. Не используйте заведомо опасные для здоровья элементы дизайна

2.3.1. Ограничение в три или менее вспышки в секунду (Level A)

•На странице не должно быть элементов, которые вспыхивают более 3 раз в секунду. Подробное описание понятий вспышки и допустимых значений можно найти в стандарте https://www.w3.org/Translations/WCAG20-ru/#general-thresholddef.

2.4. Навигация: предоставьте пользователям помощь и поддержку в навигации, поиске контента и в определении их текущего положения на сайте

2.4.1. Пропуск повторяющихся на всех страницах блоков (Level A)

•Имеется ссылка для пропуска повторяющихся блоков.

•Имеется корректная иерархия заголовков начиная с уровня 1.

2.4.2. Заголовок страницы (Level A)

•Страница имеет информативный заголовок <title>, описывающий её предназначение и цели.

•При навигации по сайту без перезагрузки страницы также следует изменять содержимое <title>.

2.4.3. Порядок перемещения фокуса (Level A)

•Порядок навигации по ссылкам, элементам формы и др. объектам интуитивно понятен и логически обоснован.

2.4.4. Предназначение ссылки (в контексте) (Level A)

•Предназначение каждой ссылки (или другого активного элемента) ясно из самого текста ссылки, либо из текста ссылки в сочетании с ее программно вычисляемым контекстом.

•Ссылки с одинаковым текстом, ведущие в разные места, легко различимы между собой.

2.4.5. Различные способы поиска (Level AA)

•Пользователю доступно более одного способа поиска нужной веб-страницы: список связанных страниц, оглавление, карта сайта, поиск, список всех страниц сайта и т.д.

2.4.6. Заголовки и метки (Level AA)

•Заголовки и метки (<label>, <legend>, aria-label) должны быть информативны и чётко описывать ту часть страницы или тот элемент, к которому они относятся.

•Желательно избегать повторяющихся заголовков и меток, если только структура страницы не предоставляет очевидного способа их различения.

2.4.7. Видимый фокус (Level AA)

•Элемент с фокусом должен быть чётко различим.

3.1. Удобочитаемость: сделайте весь текстовый контент удобочитаемым и понятным

3.1.1. Язык страницы (Level A)

•Язык страницы необходимо указывать с помощью атрибута <html lang>.

3.1.2. Язык частей страницы (Level AA)

•Для элементов, язык которых отличается от языка страницы, его также необходимо указывать с помощью атрибута lang.

3.2. Предсказуемость отображения и функционала

3.2.1. Предсказуемость при фокусе (Level A)

•Попадание фокуса на какой-либо элемент не должно вызывать значимых изменений на странице (смены контекста):

•переход на другую страницу;
•открытие всплывающего окна;
•потеря или непредсказуемое повторное изменение фокуса;
•непредсказуемая перемотка страницы;
•другие изменения, которые могут запутать или сбить с толку пользователя.

•Такие вещи, как развертка списка, показ динамического меню или переключение табуляторного элемента управления, допустимы. Подробнее см. стандарт WCAG 2.0.

3.2.2. Предсказуемость при вводе (Level A)

•Ввод информации или взаимодействие с каким-либо полем или элементом управления не должен вызывать значимых изменений на странице (смены контекста), если пользователь не был проинформирован об этом заранее.

Примеры смены контекста можно найти выше в п. 3.2.1.

•Наиболее распространенный случай - переход на другую страницу или открытие нового окна по событию onchange на элементах <select>, радиокнопках или чекбоксах. Следует по возможности избегать такого поведения, переходить на страницу только по нажатию на кнопку или на ENTER (G80 <g80_>_).

При тестировании следует учесть, что событие onchange в разных браузерах срабатывает по-разному.

3.2.3. Единообразная навигация (Level AA)

•Навигационные элементы, представленные на нескольких страницах сайта, при навигации по сайту не должны менять порядок и расположение.

3.2.4. Единообразие названий (Level AA)

•Элементы со схожей функциональностью на разных страницах должны быть подписаны схожим образом.

3.3. Помощь при вводе: помогайте пользователям избегать ошибок при вводе информации и исправлять их

3.3.1. Выявление ошибок (Level A)

•Обязательные поля размечены соответствующим образом для зрячих и незрячих: упоминание об обязательности поля в <label>, атрибут aria-required на поле ввода.

•Ошибки формы дожны быть представлены в интуитивном и доступном виде.

В большинстве случаев рекомендуется следующий подход: у элемента с ошибкой должен быть id, по которому на него поле ссылается с помощью атрибута aria-describedby.

Атрибут aria-describedby меняется динамически: в обычном состоянии он указывает на подсказку или описание поля, а в случае ошибки ввода - на текст ошибки (вместо или вместе с подсказкой - aria-describedby может содержать несколько идентификаторов).

•Желательно предоставить общее сообщение об ошибке в начале страницы. Плюсом будет также возможность перехода из сообщения по ссылке к первому полю с ошибкой.

Во всех деталях данная техника описана в статье от WebAIM.

•Чтобы немедленно зачитать ошибку пользователю в некоторых случаях можно использовать уведомление с помощью role="alert" или aria-live (ARIA21 <aria19_>_).

3.3.2. Текстовые метки и инструкции для полей ввода (Level A)

•Имеются информативные и правильно связанные метки (<label>, aria-label, aria-labelledby), подсказки и инструкции (aria-describedby), примеры заполнения полей, заголовки группировок полей (<legend>).

•Подсказки и инструкции к полям ввода и ссылкам, связанные с помощью aria-describedby (ARIA1, OAA44).
•Обозначение обязательных полей с помощью aria-required или required (ARIA2).

•Обозначение полей с ошибкой с помощью aria-invalid (ARIA21 <aria21_>_).

Следует обратить внимание на поля состоящие из нескольких элементов (например, радиокнопки, выбор даты - от и до, телефон - отдельно код и номер и т.д.): необходимо синтактически связать общую текстовую метку с каждым полем. Например, <fieldset> и <legend>, либо aria-labelledby="field-label-id subfield-label-id", либо aria-label="Дата от".

3.3.3. Подсказки при ошибках (Level AA)

•При ошибках в заполнении формы, найденных на клиентской или серверной стороне, когда это уместно, нужно выводить в доступном виде подсказки и рекомендации по их исправлению.

3.3.4. Предотвращение ошибок (для юридических, финансовых и пользовательских данных) (Level AA)

Для ввода данных, которые имеют юридическую или финансовую значимость, для других контролируемых пользователем данных (определение дано в стандарте), выполняется **хотя бы одно** из требований:

•Проверка. Данные, введенные пользователем, проверяются на наличие ошибок ввода, и пользователю предоставляется возможность исправить ошибки.

•Подтверждение. Предоставлен механизм для подтверждения и исправления информации перед окончательной отправкой данных. Например, запрос подтверждения действия или предпросмотр введённых данных.

•Обратимость (если возможно). Отправленные данные можно вернуть, например, в течение заданного промежутка времени.

4.1. Максимальная совместимость с существующими и будущими приложениями, включая ассистивные технологии

4.1.1. Синтаксис (Level A)

•Отсутствуют существенные ошибки валидации HTML/XHTML (http://validator.w3.org/)

Максимальное соответствие спецификации используемой версии HTML не является обязательным, но рекомендуется (см. технику G192).

Важные для доступности сайта моменты:

•повторяющиеся идентификаторы (id) элементов (ошибка 77),
•ошибки использования открывающих и закрывающих тегов (ошибка 70),
•опечатки в именах тегов и атрибутов,
•несоблюдение формата машиночитаемых данных (например, ошибки в URL, в значениях тега <time> и атрибута datetime).
•другие ошибки, если они могут исказить восприятие сайта браузерами и экранными чтецами.

4.1.2. Заданы имена, роли, значения, состояния и взаимосвязи между элементами (Level A)

•Разметка должна корректно восприниматься программами доступа.

•Контролируемые браузером состояния (managed state) - фокус и выделение текста - должны быть заданы корректно в каждый момент времени. Это может быть либо встроенный механизм фокуса, либо один из описанных здесь: Providing Keyboard Focus.

•Если элементы имеют роли, значения, названия, или между ними есть отношения, они должны быть выражены по возможности с помощью role, aria-* и других атрибутов.

Ниже неполный список случаев, когда уместна дополнительная семантическая разметка:

•Предоставление текстовых меток к полям, ссылкам и управляющим элементам, подписей к изображениям (ARIA6 - ARIA10, ARIA14 - ARIA16).

•Разметка структуры страницы: семантических областей, заголовков (ARIA11 - ARIA13, ARIA20).

•Группировка полей формы в role="group" - аналог <fieldset> (ARIA17).

•Зачитываемые пользователям ошибки и уведомления (ARIA18, ARIA19).

•Отмена семантики элемента с помощью role="presentation" (например, чтобы используемая для вёрстки таблица не воспринималась как таблица с данными).

•Разметка индикаторов загрузки role=progressbar (OAA27).

•Разметка открывающихся поверх страницы модальных окон с помощью role="dialog" (OAA2, WAI ARIA Authoring practices 3.3).

•Разметка вкладок или аккордеона с помощью role="tablist". (OAA34, OAA35, OAA36, OAA37).

•Разметка собственной реализации комбинированного списка с помощью role="combobox". (OAA9, OAA12).

•Разметка поля ввода с автодополнением role="combobox" и aria-autocomplete="list" (OAA11, OAA14).

•Разметка ссылки, раскрывающей или скрывающей какой-либо блок на странице с помощью aria-expanded (w3c wiki).

Внимание! Пример на Open Ajax Accessibility содержит ошибку и противоречит примеру на w3c wiki испецификации WAI-ARIA, не следует на него ссылаться или использовать. Атрибут aria-expanded должен быть не у раскрывающегося элемента, а у связанной кнопки или ссылки.

•Разметка интерактивного меню со вложенными элементами (OAA26, OAA27, WAI ARIA Authoring practices 4.4).

•Разметка слайдера (выбора значения в диапазоне) с помощью role="slider" (OAA32).

•Разметка деревьев с помощью role="tree" (OAA41, OAA42, OAA43).

•Скрытие элементов с помощью aria-hidden="true".