HYPERMEDIA APPLICATIONS (WEB AND MULTIMEDIA)

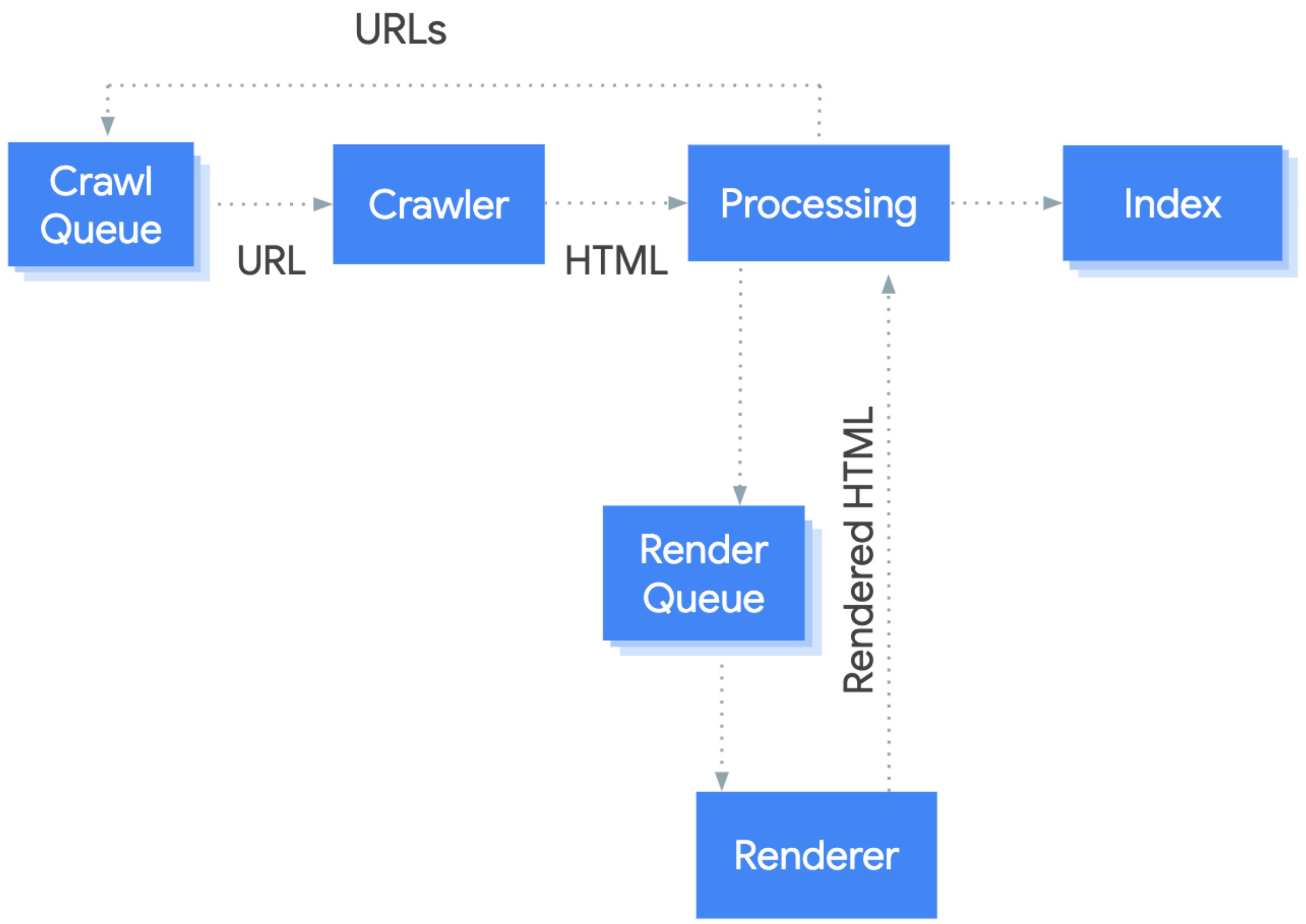
SEO & Accessibility

SEO – What? (again)

Search Engine Optimization (**SEO**) is the process of making your website better for search engines (aka Google).

Your website is given a score based on some known and some unknown parameters evaluated by the crawlers.

SEO – How?


URLs
Crawl Queue
URL
Crawler
HTML
Processing
Index
Render Queue
Rendered HTML
Renderer

SEO – Why?

SEO – so far

The important thing for the crawler is that the page is complete and filled with the content the moment it’s going to be analysed.

Nuxt allows to achieve that easily by changing the rendering of the project.

SEO – head

HTML can help us improve our website thanks to the <head> tag.

The head tag contains all the information about the page and the metadata that allow to describe its content, making it understandable to the crawler.

SEO – head

Some of the information that can be put in the head tag are:

- **Title →** Useful to give a short and insightful description of the page
- **Description →** The actual description of the webpage
- **Keywords →** The list of keywords for search engines
- **Character set →** What characters are permitted on the webpage

And many others.

SEO – NuxtJS

Problem 1: NuxtJS is a framework that creates and manages a SPA. How can we distinguish the pages?

Problem 2: NuxtJS uses SFC instead of HTML files. Where do we put the head tag (and thus, the information)?

SEO – NuxtJS

Nuxt offers two ways to add the metadata to our website:

- **Globally** → in the nuxt.config.ts file
- **Locally** → in the script or template section of each **page**

SEO – Global

```
export default defineNuxtConfig({
    app: {
        head: {
            title: "My Page title",
            charset: "utf-8",
            meta: [
                {
                    name: "description",
                    content: "This is the description of the webpage"
                },
                {
                    name: "keywords",
                    content: "key1, key2"
                }

                // Other metadata
            }
        }
    }
})
```

Global Head dependencies

```
export default defineNuxtConfig({
  app: {
    head: {
      script: [
        { src: 'https://awesome-lib.js' }
      ],
      link: [
        // <link rel="stylesheet" href="https://myawesome-lib.css">
        { rel: 'stylesheet', href: 'https://awesome-lib.css' }
      ]
    }
  }
})
```

SEO – Local

To have custom (and “reactive”) values for the metadata it is possible to set up the <head> tag inside each page:

- useHead({ … })
- useSEOMeta({ … })
- Using special tags in the template section

SEO – Local – Script

The main difference is the object passed to the composable function:

- “useHead()” accept an object similar to the one used for the global metadata (title, meta …)

- “useSEOMeta()” accept an object where the each metadata is placed as the field of the object

SEO – Local

```
<script setup>
  useHead({
    title: "My second page",
    meta: [
      {
        name: "description",
        content: "The description of the second page"
      }
    ]
  })
</script>
```

```
<script setup>
  useSEOMeta({
    title: "My second page",
    description: "The description of the second page"
  })
</script>
```

SEO – Local

Passing a value is done normally.

It is recommended to use getters (() => value)

```
<script setup>
  const description = ref('My amazing site.')
  useHead({
    meta: [
      { name: 'description', content: description }
    ],
  })
</script>
```

```
<script setup>
  const description = ref('My amazing site.')
  useSeoMeta({
    description
  })
</script>
```

SEO – Local – Template

There is no indication about Options API for what concerns the script.

Both “useHead()” and “useSEOMeta()” **should** work properly when called inside the asyncData().

The alternative is to use some special tags directly inside the template section of the SFC file.

SEO – Local – Template

Nuxt provides components so that you can interact directly with your metadata within your component's template.

- <Head>
    - <Title>
    - <Style>
    - <Meta>
    - <Link>

SEO – Local

```
<script setup>
const title = ref('Hello World')
</script>

<template>
  <div>
    <Head>
      <Title>{{ title }}</Title>
      <Meta name="description" :content="title" />
    </Head>

    <h1>{{ title }}</h1>
  </div>
</template>
```

SEO – some known rules

Title

- Use a brief and accurate description of the page’s content
- Create unique titles for each page of your website

Description

- Accurately summarize the page content

Headings

- Imagine you’re writing an outline

Responsiveness

- If you are using Responsive Web Design, use the <meta name="viewport”> tag to tell the browser how to adjust the content.

Web Accessibility

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”

Tim Berners-Lee, W3C Director and inventor of the World Wide Web

Web Accessibility

The Web is fundamentally designed to work for all people.

When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.

Making the web accessible benefits individuals, businesses, and society. International web standards define what is needed for accessibility.

Web Accessibility – Standards

The **World Wide Web Consortium** (**W3C**) and the **Web Accessibility Initiative** (**WAI**) developed technical specifications, guidelines, techniques and supporting resources that describe accessibility solutions such as:

- Authoring Tool Accessibility Guidelines (**ATAG**)
- Web Content Accessibility Guidelines (**WCAG**)
- User Agent Accessibility Guidelines (**UAAG**)

Web Accessibility – POUR

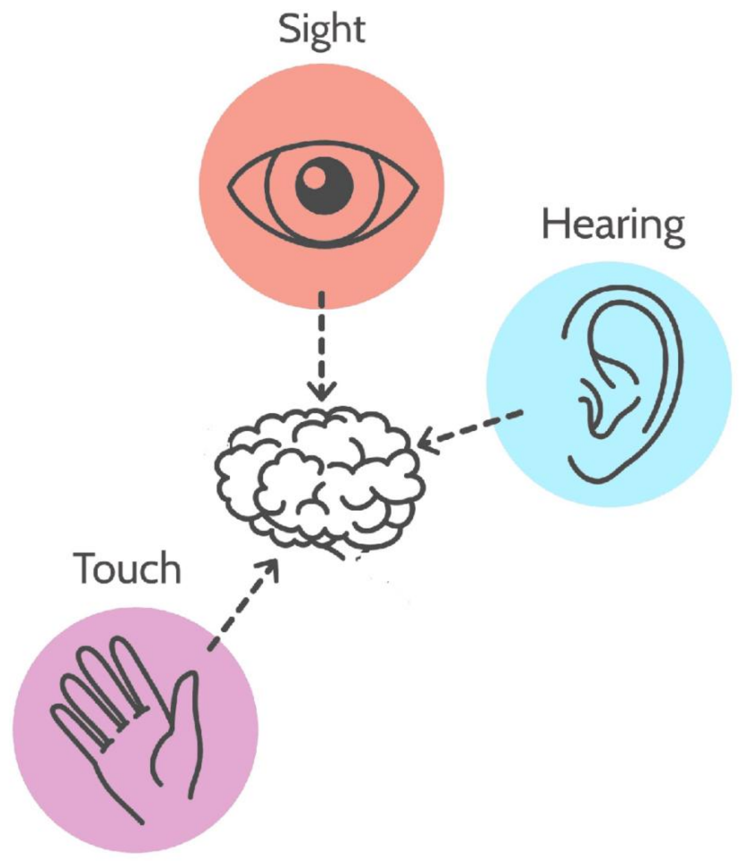
Overall, accessibility guidelines and techniques are based on four core principles:

- **Perceivable**
- **Operable**
- **Understandable**
- **Robust**

Web Accessibility – Perceivable

Perceivable means the user can **identify** content and interface elements by means of the senses.

For many users, this means perceiving a system primarily **visually**, while for others, perceivability may be a matter of **sound** or **touch**.



Web Accessibility – Perceivable

- Non-text content should have an **appropriate text alternative**
- Media (audio, video) should provide **captions**
- The content can be **presented in different ways**
- Content should be **easy to see** (and hear)

Web Accessibility – Operable

Operable means that a user can **successfully use controls**, buttons, navigation, and other necessary interactive elements.

This means that user should be able to use a **mouse**, a **keyboard**, the **touchscreen** or **any other device** to navigate the webpage.

Web Accessibility – Operable

- Functionalities are available from a **keyboard** and other different **input modalities**

- Users can easily **navigate, find content, and determine where the are**

- Content **does not cause** seizures and physical reactions

Web Accessibility – Understandable

Understandable means that people must be able to understand:

- the information in your web content
- how to **navigate** the website to **find the information** they want
- **how to use** any tools or features built into the website.

Web Accessibility – Understandable

- Content appears and behaves in **predictable** ways
- Users are helped to **avoid** and **correct** mistakes
- **Supplemental** representation of information (summaries, additional description of graphs, …)

Web Accessibility – Robust

Robust means that the website **supports** and can be **accessed** on a **variety of devices**, including assistive technologies.

When technology evolves, the content must still remain **accessible**, and be able to be **interpreted**.

Web Accessibility – Robust

- Validate against **technical standards** for any applicable platforms

- Provide a **responsive** solution to **adapt** to different devices

Accessibility – Headers

Headers are an effective way to break up large chunks of text.

People tend to used headers based on the visual appearance instead of nesting headers by rank.

According to the W3C, skipping heading ranks can be confusing to people who rely on screen readers and other assistive technology and should be avoided whenever possible.

Source

Accessibility – Alt text

Alternative text is a written description of an image that screen readers can read out loud for people with visual impairments, sensory processing disorders, or learning disorders.

According to the W3C, all images must have a text alternative that describes its function or the information it represents.

However, that doesn’t mean every image should be treated the same.

Source

Accessibility – Alt text

- **Informative images**
  - a picture intended to visually represent an important concept or information
  - the text alternative should be a short description conveying the essential information presented by the image.

- **Decorative images**:
  - an image whose only purpose is to add visual decoration
  - adding a null (empty) text alternative (alt=“”) tells assistive technologies like screen readers to ignore, or not announce, the image.

Source

Accessibility – Alt text

- **Functional images**:
  - an image used as a link or button (such as a printer icon to represent the print function)
  - the text alternative should describe the functionality of the link or button, rather than the visual image.

- **Images of text**:
  - If readable text is presented within an image, the text alternative should contain the same words as the image.

Source

Accessibility – Alt text

- **Complex images**:
  - Images used to convey data or detailed information (such as a graph or diagram)
  - the text alternative should be a complete text equivalent of the data or information provided in the image.

Source

Accessibility – Decorative Images

- The information provided by the image might already be given using adjacent text
- the image might be included to make the website more visually attractive.

A way to decide whether an image is decorative is to pretend to read your web page out loud to someone over the phone. If the image doesn't contain any information and isn't a link or button, you can treat it as decorative.

Source

Accessibility – Alt text

A few useful links:

- https://webaim.org/techniques/alttext/
- https://supercooldesign.co.uk/blog/how-to-write-good-alt-text
- https://www.w3.org/WAI/tutorials/images/

Source

Accessibility - Colours

When deciding on the colour scheme of your page, use a combination that allows every part of your website to be readable.

THIS IS MY CONTENT

THIS IS MY CONTENT

THIS IS MY CONTENT

THIS IS MY CONTENT

WAI – ARIA

Accessible Rich Internet Applications (ARIA) is a set of **roles** and **attributes** that define ways to make web content and web applications more accessible to people with disabilities.

ARIA provides a **framework** for adding attributes to **identify features** for user interaction, how they **relate** to each other and their current **state**.

WAI – ARIA

ARIA provides the following:

- **Roles →** describe the widget or the structure of the web page
- **Properties →** describe the state of the widget and what parts of the page are likely to get updates

List of roles and properties

WAI – ARIA

```
<div id="percent-loaded" role="progressbar" aria-valuenow="75"
     aria-valuemin="0" aria-valuemax="100">
  ...
</div>
```

WAI – ARIA

From HTML5 some of the functionalities of ARIA are already implemented inside the **semantic** tags.

It is better to use them instead of repurposing something else.

The rule of thumb is:

“No ARIA is better than bad ARIA”

WAI – ARIA

```html
<progress id="percent-loaded" value="75" max="100">75 %</progress>
```

Web Accessibility – How to check

If you want to check the accessibility score of a website you can use:

WAVE

web accessibility evaluation tool

Lighthouse