Content Creation Considerations

This document is intended to serve as an overview to the type of considerations that are important to have when creating web content in general. It is always important to align expectations with your client as they may have individual requirements.

Few important clarifications for non-technical users:

HTML
HTML is the markup language used to define the structure of web content, e.g. headers, paragraphs, links etc.

CSS
Cascading StyleSheets is the technology that enables the developer to apply design to web elements. Can be referenced in a separate file or inline with the HTML.

JS
JavaScript is the technology that provides interaction to web content.

Introduction
There are three key aspects when creating new content for web platform:

- Browser compatibility
- Performance
- Resolution

Below we will look into

Browser Compatibility
It's important to set a baseline for where the content needs to work. This decision will have great impact on content strategy. For interactive content, as most presentations are, a graded support is the recommended strategy but make sure to align expectations with your client. It's still necessary to make decisions what is supported where though.

A graded support strategy means that you divide the browser baseline into groups:

- A-grade browser: All features and design are supported
- C-grade browser: All content should be accessible, but not interactivity and design
- X-grade browser: Unknown/rare browsers. Assumed to be modern and capable

Some important considerations when choosing the baseline:

- Current global browser usage
- Current local browser usage
- Target audience
- Content type

Current Global Browser Usage

This is what's normally used to create the general recommendations that can be found on the web. You need to decide where the cut-off is; should you for example support a browser with 5% or less of user share?

Current Local Browser Usage

Often content is published for specific regions, e.g. Europe or Asia. The browser usage statistics can vary greatly between areas and this will then provide better support for the decision.

Target Audience

The baseline will naturally differ greatly if the content is intended for use within a company or to the general public. It might also differ if it's only intended to professionals in a field or to any user.

Content Type

Content that is mostly about providing information (i.e. text, images, videos, etc.) is generally easy to provide fallback solutions to, e.g. if certain design elements are not supported in older browsers.
Highly interactive content, i.e. web applications, are very difficult or sometimes impossible to provide an alternative to.

Performance

There are two types of performance to consider: load and run-time. The one that we can control the most is the load performance and it's what we refer to here.
Load performance is the time it takes from opening a presentation until it's fully interactive. The less code and assets you load initially, the less time it will take before the content can be used.

Two major strategies are considered when optimizing load time:

- Size of resources referenced
- Number of network requests

Size

The overall size (usually referred to in kilobytes) can be controlled by selective loading and code minification.

Selective Loading

It's possible to load only what's necessary to render the initial content, and then load assets and code resources as they are needed.

Minification

When creating code a lot of the size is made up of empty spaces and comments in the code. There are many tools available today that will remove all spaces and comments for production. To lower the size even more it's also possible to obfuscate the code, which means that variables and other names used in the code are replaced with single letters.
Minification and obfuscation should only be done for production release as it's virtually impossible to debug.

Requests

When accessing online content, each link to a file (e.g. CSS or JS) will generate a network request to the server. The number of active requests is limited and often they are loaded synchronously, meaning that one needs to load before the other.
There are, again, two considerations to take:

- Order of loading
- Number of links

Order

In what order the files are loaded can have a great impact on the **perceived** loading time. The order will not affect the actual time it takes to load the presentation but it will affect when certain features are available. The recommended practice is to first load CSS (in the head of the document), then the initial content (HTML), and lastly the JavaScript at the bottom of the document.
This will ensure that the styled content is visible as soon as possible, i.e. before the JavaScript is loaded. As it's usually the JavaScript that makes up the bulk of the size, it can have major impact of the perceived load time.

Number of Links

In order to reduce the number of network requests, the best practice is to combine the CSS and JavaScript into as few files as

possible. Often it means that there is overall two network requests (one for CSS and one for JS) plus whatever is needed for media resources (videos, images, etc.) referenced in the initial content.

Resolution

Web content can today be viewed in a large number of devices, from small phones to large TV-screens with 4K resolution. This pose a difficult challenge for developers but it's at the same time one of the great strengths of web content.
Again, it's important to beforehand have a clear idea how the content is intended to be used in order to form a good strategy for handling varying resolutions.

In general there are two strategies to consider when creating content for multiple resolutions:

- Device/Resolution specific content
- Responsive Design

Device/Resolution specific content

With this strategy different content is prepared for different devices/resolutions. Between 1 to 3 different outputs are generally considered, but it can vary greatly per project. Examples are mobile, web and large screens (TVs, projectors etc).
This strategy is usually considered when existing content need to be adapted. However, even if it might seem to be a cheap solution initially it usually ends up being more expensive than responsive design as maintenance and updates (i.e. adding another device/resolution) is more difficult.

Advantages

- Easier to optimize code sent to each device
- Usually cheaper in initially with existing content

Drawbacks

- Content often end up in different states for the various devices
- Multiple code bases is more difficult and expensive to maintain/update
- Harder to test
- Often end up adopting mechanisms of responsive design anyway to serve slightly different designs to related devices (e.g. normal-size vs big-size smartphones).

Responsive Design

Responsive Design is a terminology used to describe web content that have been developed to adjust to various resolutions. With this strategy, a few breakpoints based on the screen width/height are

chosen and then the content is developed to look and behave well at any resolution. Examples of breakpoints (not recommendations, just examples):

- Width of 480 pixels or less
- Width between 481 and 1024 pixels
- Width of 1025 pixels or more

The important point here is: **same content is served to all devices** and it is therefore easier (less expensive) to update and maintain. CSS can be used to hide content in any resolution if that is desired (e.g. hide some interactivity that does not work well on a mobile phone).
To use Responsive Design successfully it is paramount to consider the breakpoints in the design phase! With existing content that need to be adopted it should go back to the designers so that they can update with designs for the new resolutions.

Advantages

- Better prepared for future devices/resolutions
- Cheaper to maintain and update
- Same content served to any user
- Works if browser is resized (means it's also easier to test)

Drawbacks

- Challenging to adopt existing content
- More difficult to optimize size of CSS delivered to device