LIFERAY

DYNAMIC USER INTERFACES

for Desktop and Mobile

Iliyan Peychev

Core Developer

Agenda

- Responsive design using JavaScript - AlloyUI Viewport
- Responsive design using CSS - Media Queries
- Dynamically retrieving data - Pjax and A.Plugin.ScrollInfo
- The future - CSS Flexible Box Layout Module

Responsive design using JavaScript

AlloyUI Viewport allows you to adapt your layout based on the following size groups:

- 320px - smart phones in portrait mode
- 480px - smart phones in landscape mode
- 720px - for tablets in portrait mode
- 960px - for Desktop browsers

Overwriting the magic numbers

These are in **"defaults.viewport"** namespace so you can overwrite them.

JavaScript

```
var viewport = YUI.AUI.namespace('defaults.viewport');

viewport.columns = {
        ...
};
```

Supports greater than or less than the specified widths.

- It is also possible to target widths, greater than or less than the specified.
- If you have a device with 600x800px screen resolution, you can still target that device with the CSS classes.

How to use it

Just add "aui-viewport" module to the list of modules on your page:

JavaScript

```
AUI().use('aui-viewport', ...);
```

How does it work

It adds a few classes to the HTML element depending on the width of the window:

HTML

```
<html class="aui-view-gt320 aui-view-gt480 aui-view-gt720 aui-view-gt960 aui-view-960">
```

Based on these classes, you may create selectors which match some devices only.

AUI Viewport example

Order the navigation items in a row for tablets...

CSS

```
#navigation li {
        display: inline;
        float: left;
}
```

AUI Viewport example

...or in column mode for smart phones

CSS

```
.aui-view-lt720 #navigation li {
        display: block;
        float: none;
}
```

Target specific browsers

CSS

```
/* Browsers on smartphones */
.touch.aui-view-lt720 {}

/* Webkit based tablets and smartphones */
.webkit.aui-view-lt960 {}

/* Smaller browser views on just Windows */
.win.aui-view-720 {}
```

Pros and cons

Pros

- Simple and powerful
- Will work on browsers which don't support Media Queries

Cons

- Will not work if JavaScript is disabled
- JavaScript blocks rendering process

Supported browsers




6+

Responsive Design using Media Queries

Responsive Design using Media Queries

Media Queries allow adapting the same content to a specific range of output devices.

Responsive Design using Media Queries

Media Queries allow adapting the same content to a specific range of output devices.

CSS3 Media Queries extend the media queries we had in HTML4 [HTML401] and CSS2 [CSS21]:

‘aural’, ‘braille’, ‘handheld’, ‘print’, ‘projection’, ‘screen’, ‘tty’, ‘tv’

Loading CSS file only if needed

HTML

```
<link media="(min-width: 40.5em)" href="extensions.css"
rel="stylesheet" />
```

HTML

```
<link media="not screen and (color)" href="example.css"
rel="stylesheet" />
```

Example

Apply a style sheet only in portrait orientation:

CSS

```
@media all and (orientation: portrait) {
....
 }
```

A CSS pixel is **not** always a device pixel

Web developers traditionally assumed a CSS pixel as a device pixel.

However, on high DPI devices (such as iPhone 4+) a CSS pixel may repsesent multiple pixels on screen.

If we set zoom magnifcation of 2x, then 1 CSS pixel would actually be represented by a 2×2 square of device pixels.

Resolving the situation

CSS

```
@media
only screen and ( -webkit-min-device-pixel-ratio:  2      ),
only screen and ( min--moz-device-pixel-ratio:     2      ),
only screen and ( -o-min-device-pixel-ratio:       2/1    ),
only screen and ( min-device-pixel-ratio:          2      ),
only screen and ( min-resolution:                  192dpi ),
only screen and ( min-resolution:                  2dppx  ) {

  /* Retina-specific stuff here */

}
```

Opera requires fractions for the device-pixel-ratio Param. Firefox 16 supports "min-resolution" using the dppx unit.

Credits to [css-tricks.com](http://www.css-tricks.com)

Supported browsers




9+

Retrieving data

Retrieving data

Pjax Utility

Pjax Utility

Progressively enhance normal links on a page.

Pjax Utility

Progressively enhance normal links on a page.

In this way clicks result in the linked content being loaded via Ajax.

How to use it

Plug the module to the content:

JavaScript

```
AUI().use('pjax', function (A) {
    new A.Pjax({container: '#content'});
});
```

Make the links Pjax-able

By default Pjax module will handle links which have ".yui3-pjax" class.

It is easy to customize this by overwriting "linkSelector" attribute of Pjax module.

How does it work

- By default the Pjax instance listens to clicks on any link on the page that has a “yui3-pjax” class.
- When a “yui3-pjax” link is clicked, its URL will be loaded via Ajax and the loaded content will be inserted into the “#content” div, replacing its existing contents.
- When the Pjax Utility makes an Ajax request to the server, it adds a special X-PJAX HTTP header to the request.

A.Plugin.ScrollInfo

Useful when you want to:

- Implement infinite scrolling
- Lazy-load content
- Display data in the same way as native applications do it

How to use it

Plug the module to a node (may be the page body):

JavaScript

```
var body = A.one('body');

body.plug(A.Plugin.ScrollInfo);

body.scrollInfo.on('scrollToBottom', function (event) {
    // Load more content when the user scrolls to the bottom of the page.
});
```

Provides useful information

Fires multiple events depending on the direction user scrolled the content:

- **scrollToBottom** – probably the most useful event
- scrollDown
- scrollLeft
- scrollRight
- scrollToLeft
- scrollToRight
- scrollToTop
- scrollUp

The future

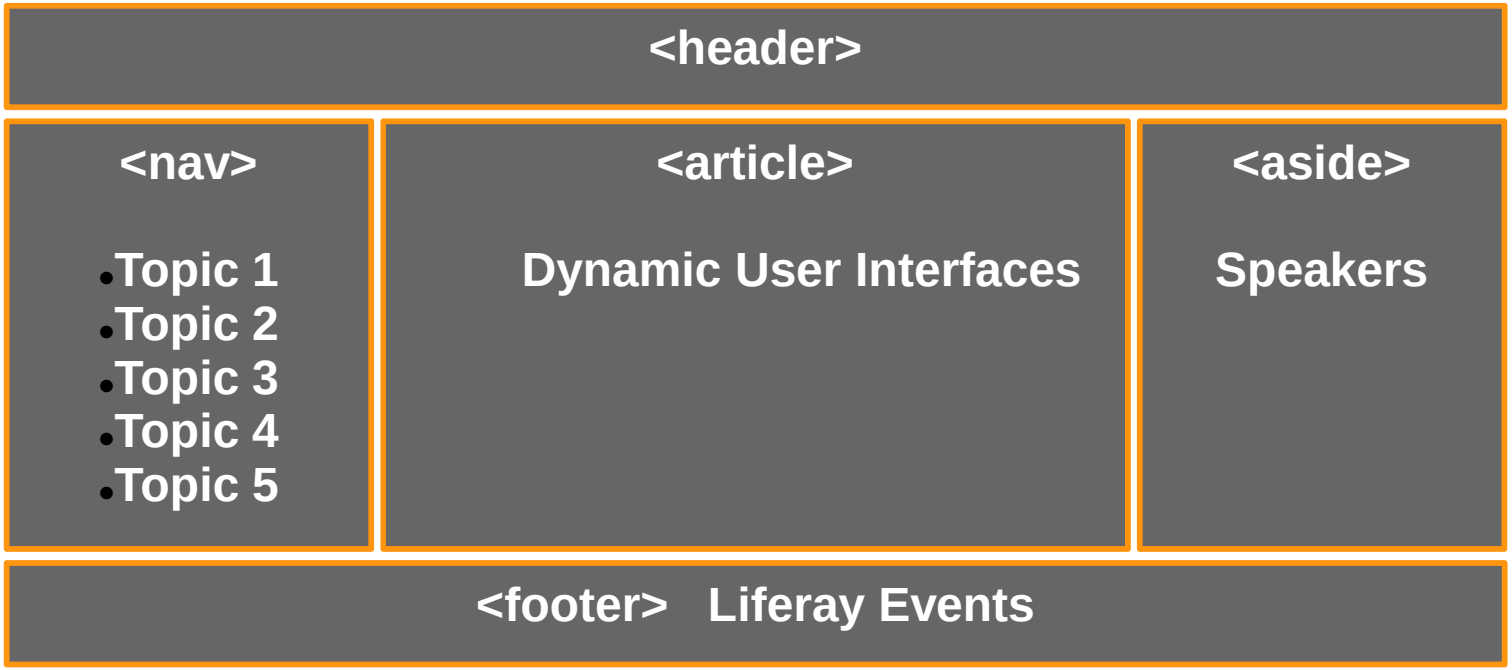
The future

CSS Flexible Box Layout Module

CSS Flexible Box Layout Module

- CSS box model optimized for user interface design.
- Similar to block layout.
- Designed for laying out more complex applications and webpages.
- Contents can be laid out in any flow direction.
- Display order can be reversed.
- Can “flex” contents sizes to respond to the available space.

The “Holy Grail Layout”

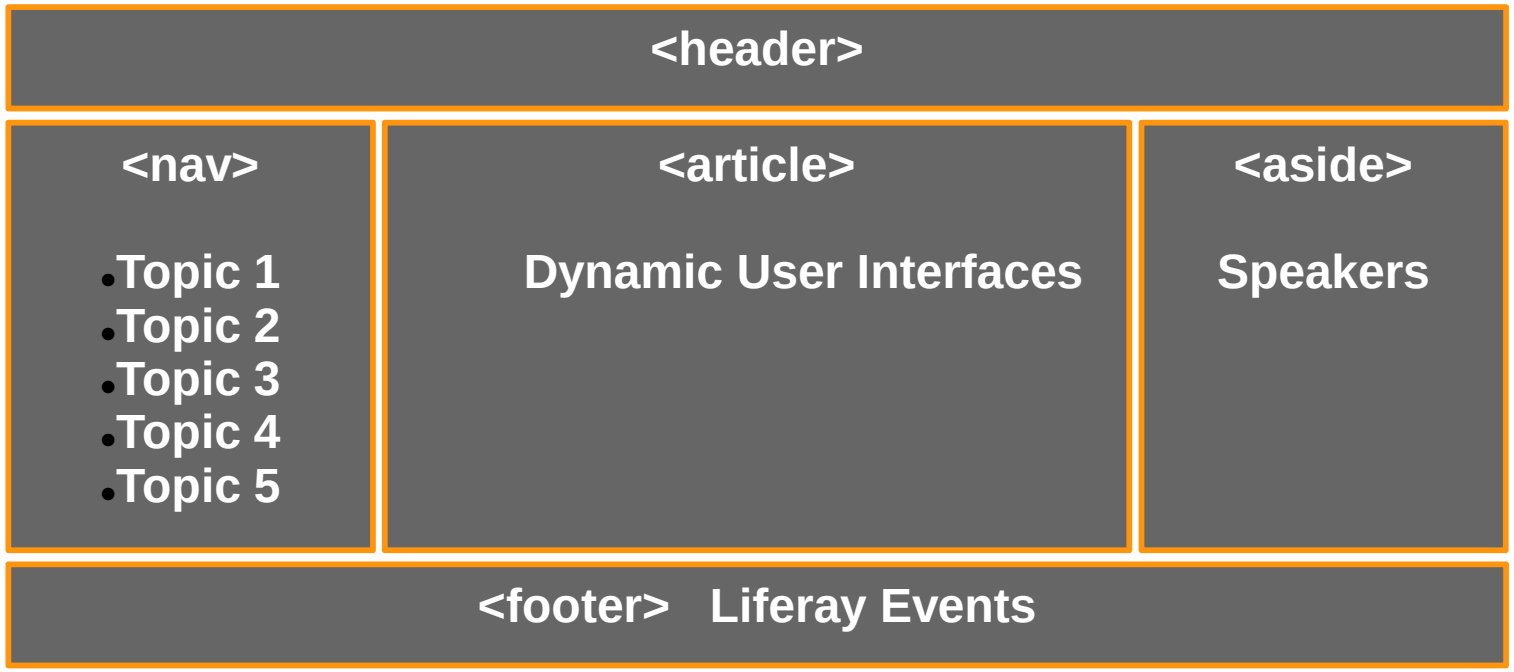


The “Holy Grail Layout” source code

HTML

```
<!DOCTYPE html>

<header>Example Header</header>

<div id="main">

        <article>Dynamic User Interfaces</article>

        <nav>Topics</nav>

        <aside>Speakers</aside>

</div>

<footer>Liferay Symposium 2012</footer>
```

The “Holy Grail Layout”



CSS

```
#main { display: flex; }
nav     { order: 1; width: 200px; }
article { order: 2; flex: 1; }
aside  { order: 3; width: 200px; }
```

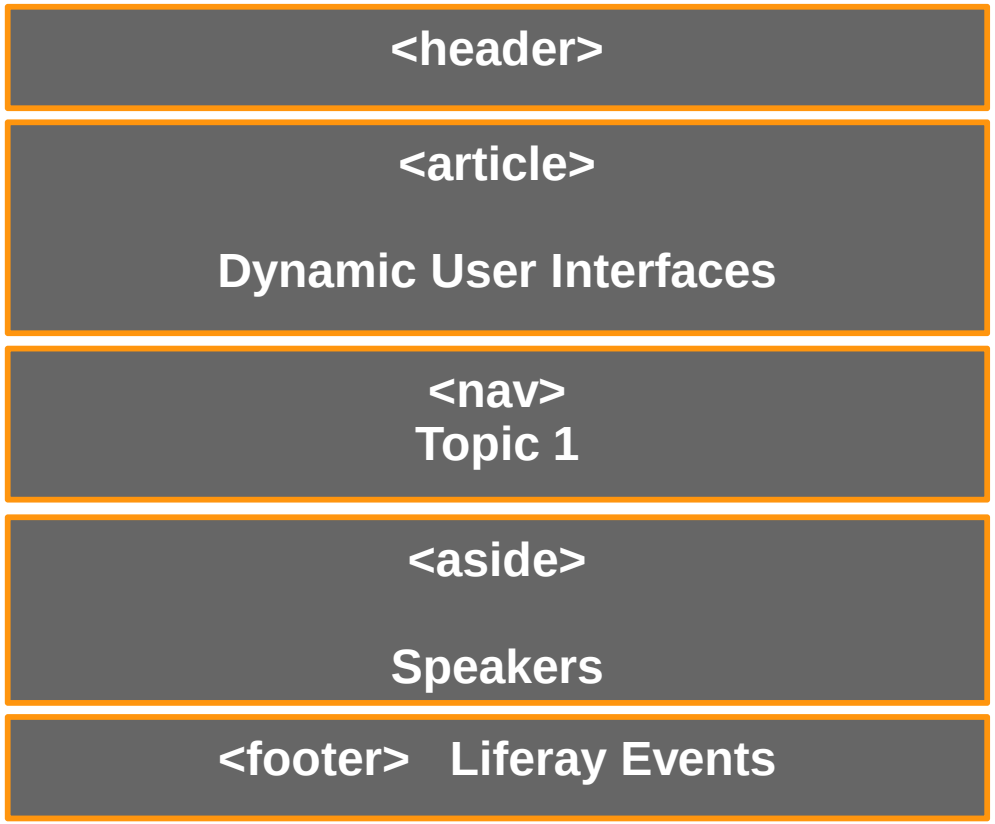
What about mobile?

Just restore document order and set the width to auto

CSS

```
@media screen and (max-width: 600px) {
        #main {
                flex-direction: column;
        }

        article {
                flex: none;
        }

        article, nav, aside {
                order: 0;
                width: auto;
        }
}
```

The “Holy Grail Layout” mobile view



Creating a header

Use “margin-left: auto” to separate flex items in “groups”.

CSS

```
.header {
        display: flex;
}

.header .login {
        margin-left: auto;
}
```

The result:

Products Services Partners **Sign In**

Switching from row to column is easy

- The contents of a flex container can be laid out in any direction and in any order.
- This functionality is exposed through the 'flex-direction', 'flex-wrap', and 'order' properties.
- Keep in mind this affects only the visual rendering and it will leave the speech order and navigation based on the source order.

Center content vertically

CSS

```
.container {
        align-items: center;
        justify-content: center;
        display: flex;
}
```

Reverse the order (row-reverse)

CSS

```
.container {
        flex-direction: row-reverse;
        ...
}
```

Reverse the order (column-reverse)

CSS

```
.container {
        flex-direction: column-reverse;
        ...
}
```

Item 3

Item 2

Item 1

Multi-line flex container

Breaks its flex items across multiple lines. The cross size of each line is the minimum size necessary to contain the flex items on the line.

CSS

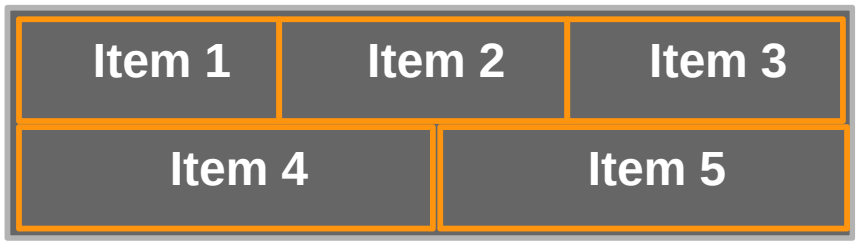
```
.container { display: flex; width: 400px; }

.multi-line .item { width: 100px;}
```

Multi-line auto flex container

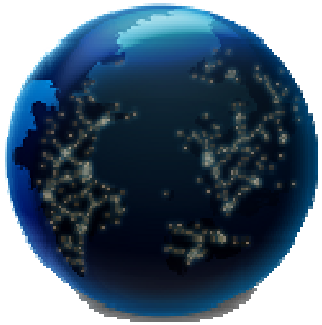
Setting "flex: auto;" to the items will force them to absorb any free space remaining.

CSS

```
.multi-line .item {
        flex: auto; /* or flex: 1 1 auto; */
        width: 100px;
}
```



Supported browsers


10+

Notes:

Firefox Nightly requires to turn on “layout.css.flexbox.enabled” in about:config

IE10 and Safari implemented an earlier draft of the specification

Summary

Adapting the layout to the device the user is currently using:

- AlloyUI Viewport (JavaScript) and Media Queries (CSS)

Summary

Adapting the layout to the device the user is currently using:

- AlloyUI Viewport (JavaScript) and Media Queries (CSS)

Retrieving the data dynamically:

- Pjax Utility and ScrollInfo plugin

Summary

Adapting the layout to the device the user is currently using:

- AlloyUI Viewport (JavaScript) and Media Queries (CSS)

Retrieving the data dynamically:

- Pjax Utility and ScrollInfo plugin

Looking at the future:

- Keep CSS Flexible Layout in your radar and stay tuned :)

Questions?

Twitter ipeychev

Google+ https://plus.google.com/101163834202763493709

email iliyan.peychev@liferay.com