
Google™
11
I/O

# Building Web Apps for Google TV

Chris Wilson and Daniels Lee
May 11, 2011

Twitter: #gtvweb #io2011

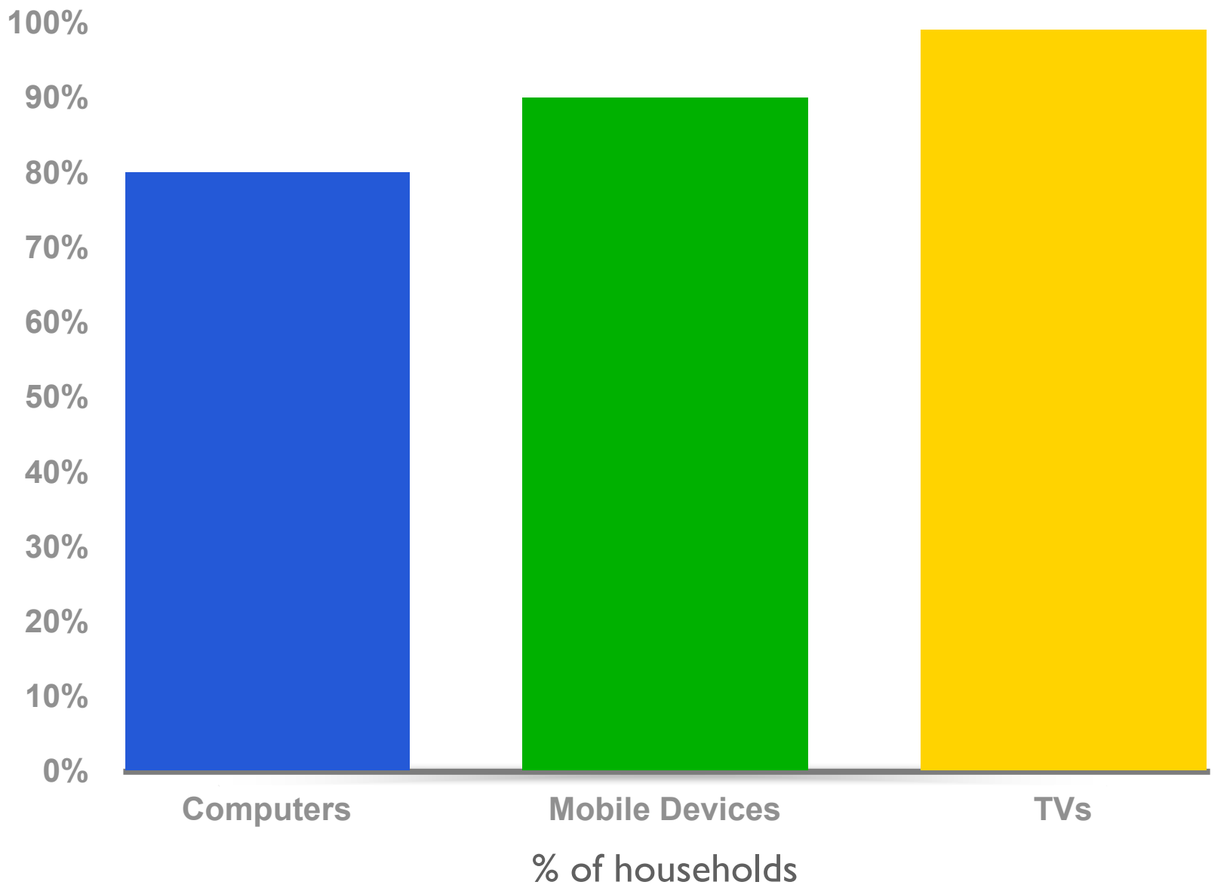

#gtvweb

# Why is Television Interesting?

#gtvweb

Google I/O 11

# Why is Television Interesting?



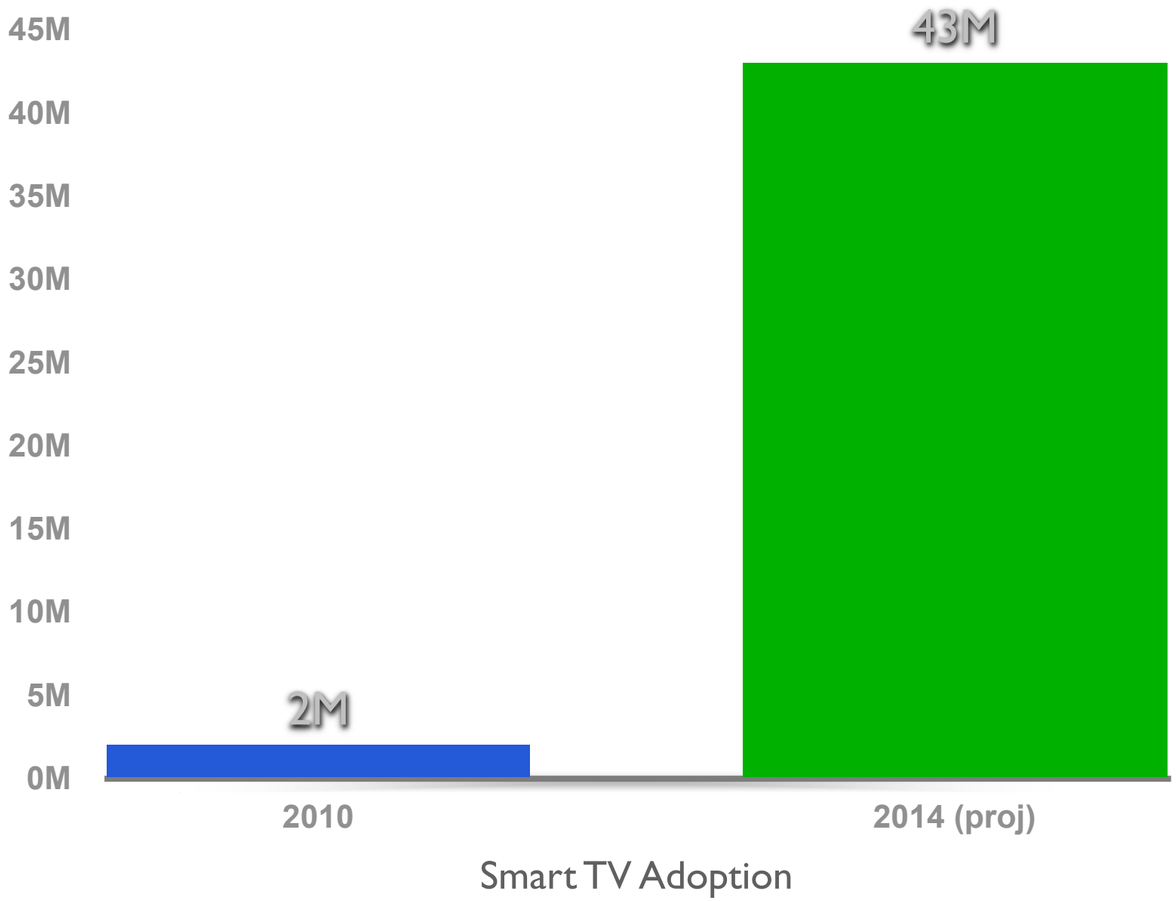

More American households have TVs than cell phones or computers.

[1] Source: Nielsen Research
[2] Source: Pew Research, Internet and American Life Project 2011
[3] Source: International Telecommunication Union, The World in 2009: ICT Facts and Figures



#gtvweb

# Why is Television Interesting?




Smart TV Adoption

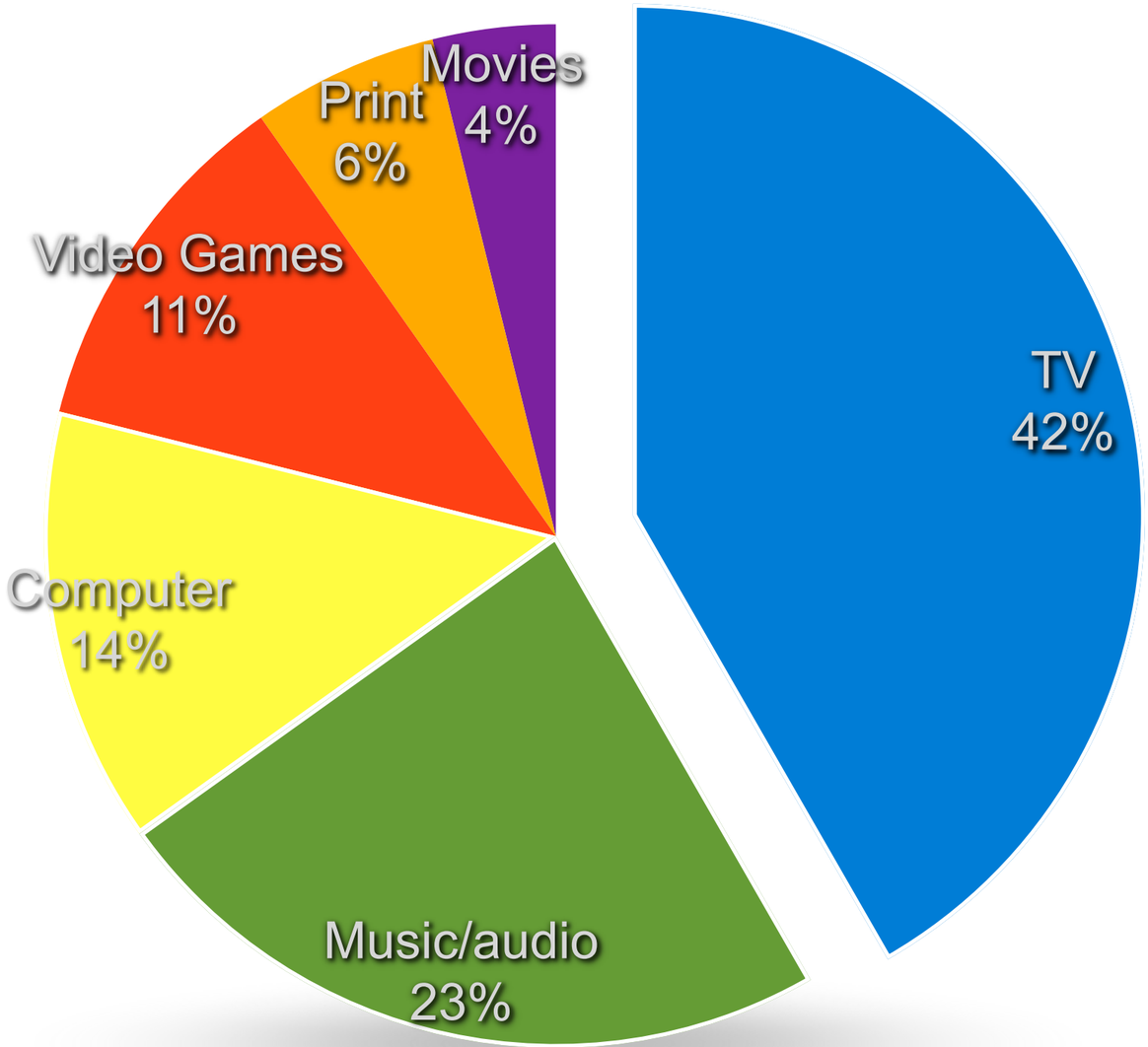
By 2014, more than 1/3 of households in the US will have an Internet-connected television.

[1] Source: James McQuivey, Forrester Research, *Connected TVs Will Sell, But Will They Get Used?, 2010*
[2] Source: In-Stat Research, *Installed Base of "Smart TV" Web-Enabled Home Consumer Electronics Devices to Reach Over 230 Million by 2014*



#gtvweb

# Media Use in 8- to 18-Year-Olds




Source: Kaiser Family Foundation: “Media in the Lives of 8- to 18-Year-Olds” (2009)



#gtvweb

# Why is Television Interesting?

- TVs are more common than computers
- Americans spend more time watching TV than any other media activity
- We should make better use of this time.

# Why Google TV?

- Make TV better
- Put great Web content on TV
- Provide an interactive platform for TV

# Opportunities

- Media consumption with social aspect
  - Watching the game “together”
- Applications that span - and morph between - devices
- Casual games, particularly multi-player

# Opportunities

- Applications using paired “virtual controllers”
  - Connecting mobile devices and tablets to “drive,” with the TV as output




#gtvweb

# Demo - TV as a social center

#gtvweb

# Google TV and the Web Platform

- Google TV is based on Android
- Google TV's web platform is Chrome, with a streamlined UI
  - "Modern Browser" (i.e. many HTML5 features, Webkit)
  - Adobe Flash (10.1) supported, supports H.264 video natively
- Updates automatically over the network


# What Makes Designing for TV Different?

TV is a different mental space:
user is in “couch mode”

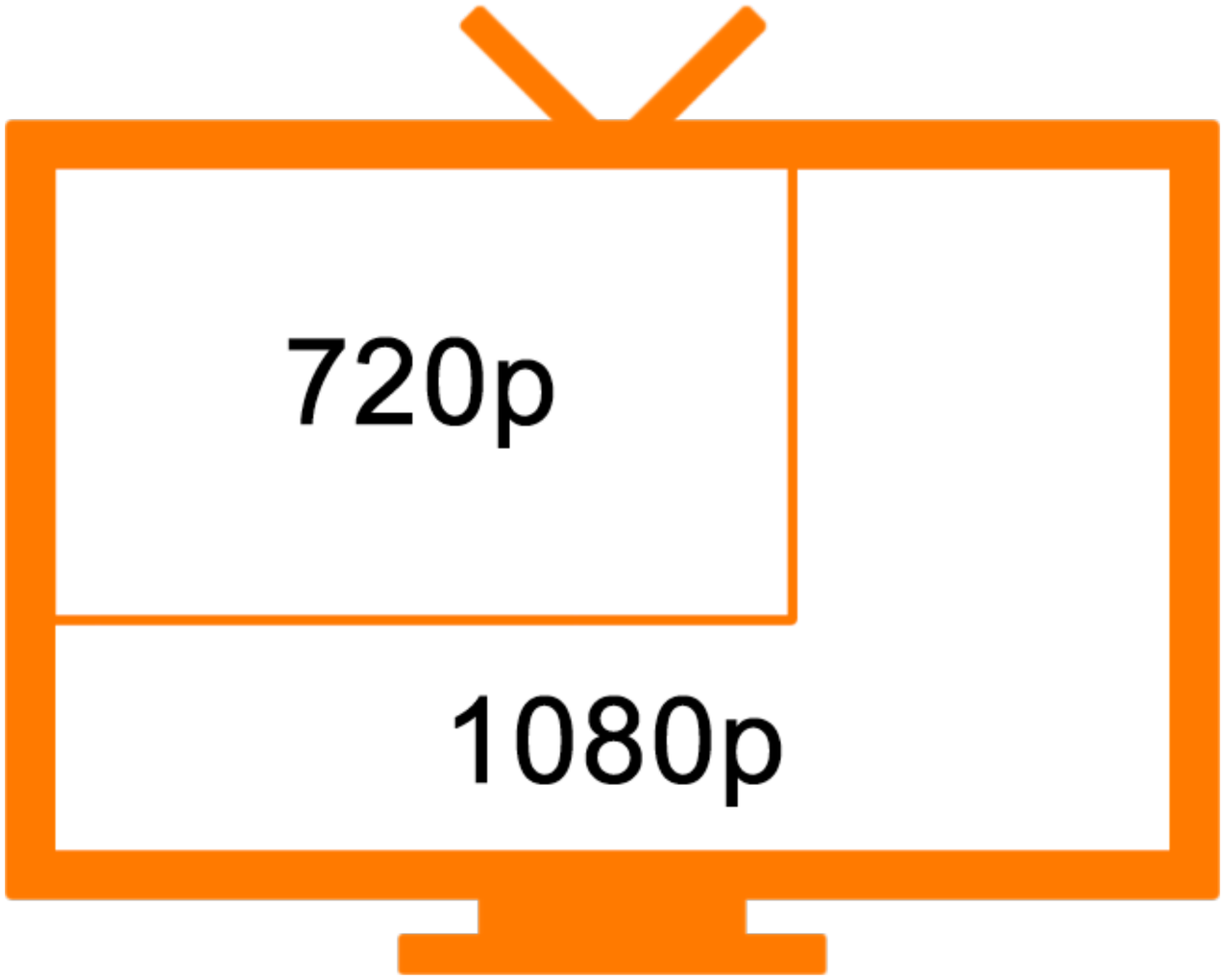
- So applications need a “10-foot UI”:
  - UI design is page-based
  - Human input is different (d-pad, not mouse)
  - TV content is dynamic - even when “static”

# Demo - “Couch Mode”

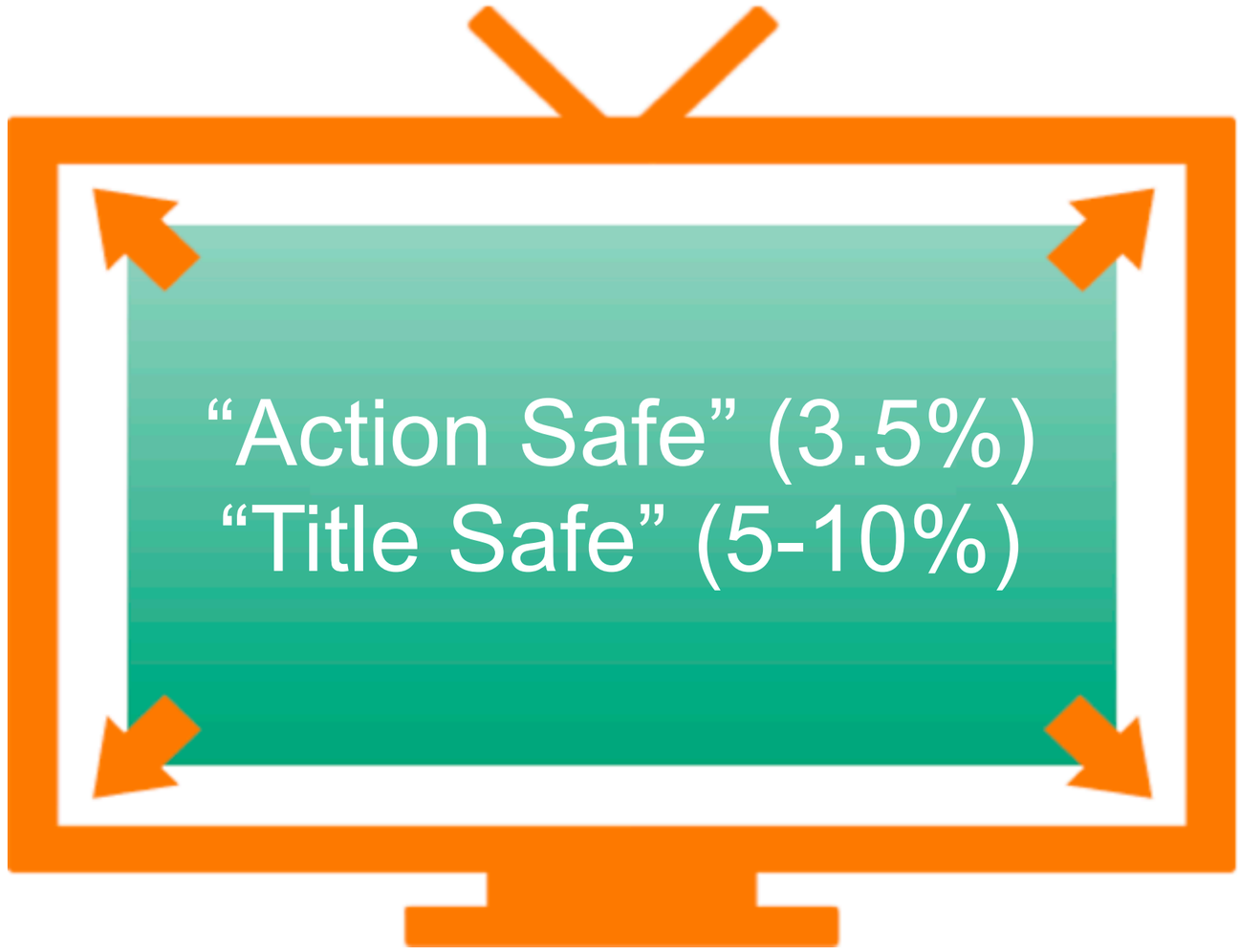
# What Makes Designing for TV Different?

- Technical constraints of TV
  - Device issues: resolution, overscan, color, etc.
- Practical constraints of TV
  - Apparent sizes, user input devices, layout, etc.
- Design guidance for TV
  - What makes a “TV experience”

# Technical Constraints - Resolution

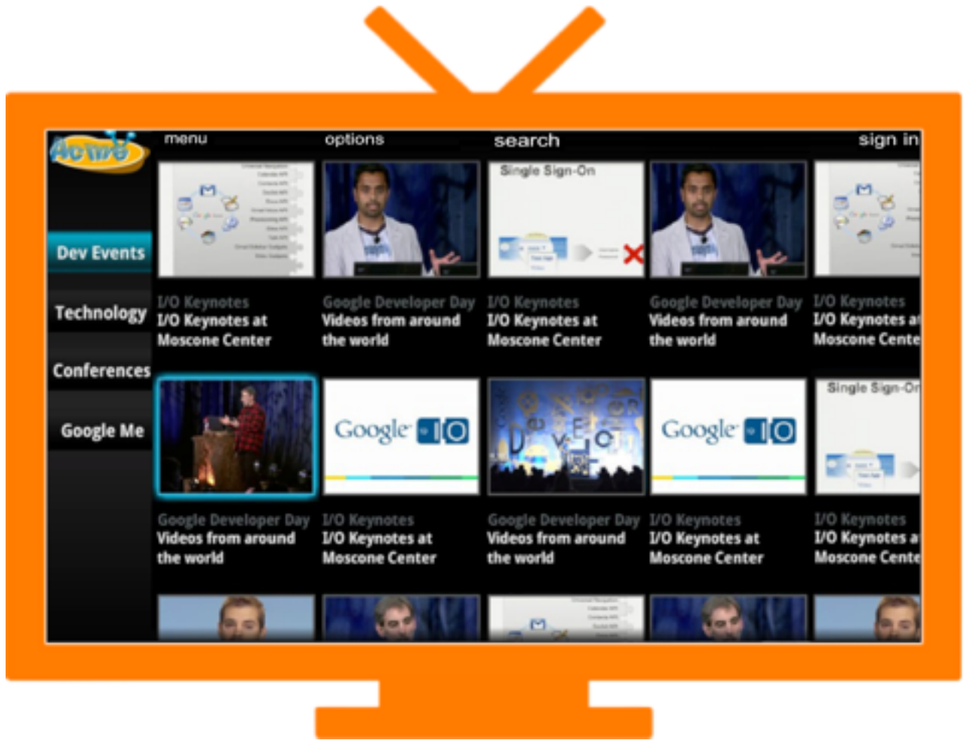


# Technical Constraints - “Overscan” and Padding



# Technical Constraints - “Overscan” and Padding

Bad



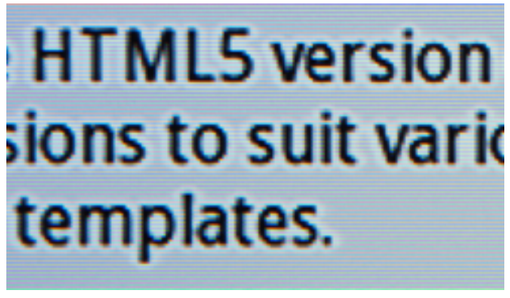

Good




#gtvweb

# Practical Constraints of Designing for TV

- The color gamut is different on TV
  - TVs are also frequently not calibrated
  - It's easy to over-saturate, so tone down your colors
    - Particularly oranges and reds
- Non-white background colors are best
  - WHITE IS LIKE ALL CAPS
  - Also, white can cause halos



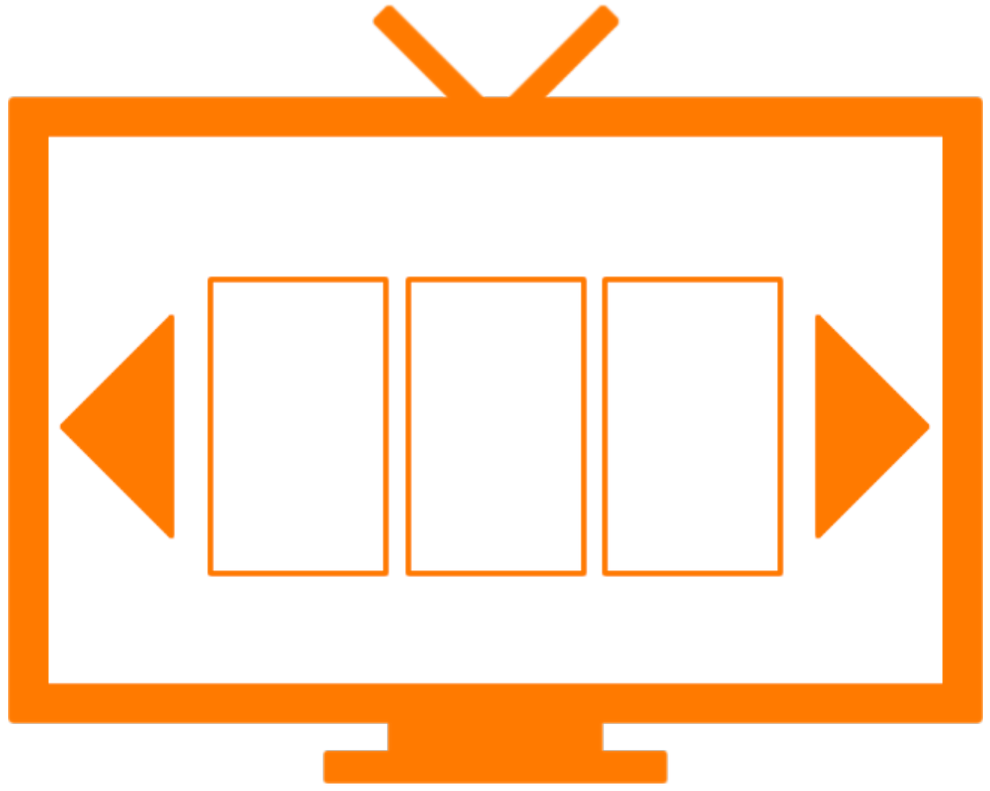

Test your contrast out on a TV display!

# Practical Constraints of Designing for TV

- Optimize your text carefully
  - Limit content length
  - Avoid small text sizes
  - Make it as big as needed, then bigger
- Typical on-screen font guidance applies:
  - Sans-serif fonts tend to be more readable
  - Don’t use too many fonts in one page
- Google TV supports font embedding

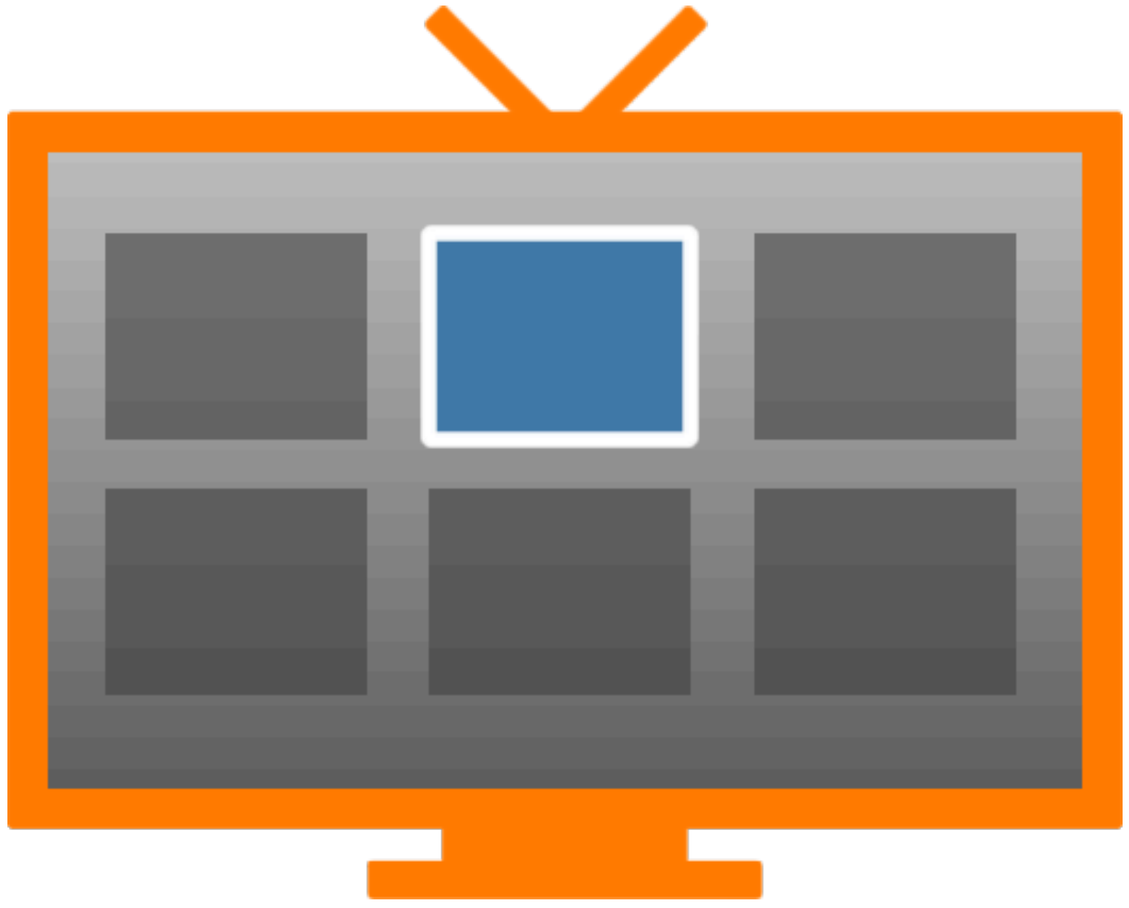

# Practical Constraints of Designing for TV

- Avoid the huge scrollable page design
- Put content “above the fold” - or give visual cues



# Practical Constraints of Designing for TV

## Have a Strong Focus Model



# Practical Constraints of TV


- User input - don't rely on a mouse, just directional-pad
  - Don't rely on keyboard shortcuts, either

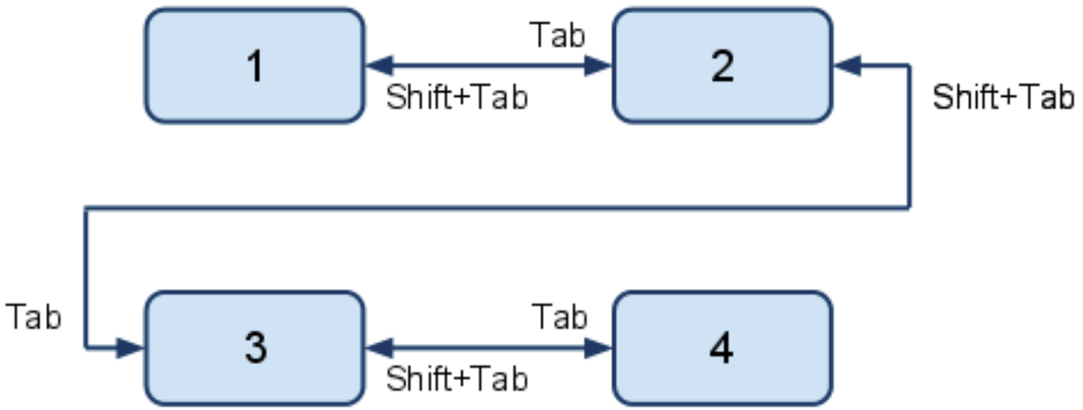
- Directional-pad navigation: it's just keypresses
  - Useful on desktop web, too: http://windowshop.com/, Google search results

```
function keydown(e)
{
    switch(e.keyCode) {
        case 37:// Left arrow - move to previous
          next = $(".selectable");
          var i = next.index( $('.selected') ) - 1;

          focus.removeClass("selected");
          if ( i < 0 )  // Error, or ran off the end- restart at end
             $('.selectable').last().addClass('selected');
           else
             next.eq(i).addClass('selected');
          break;
```

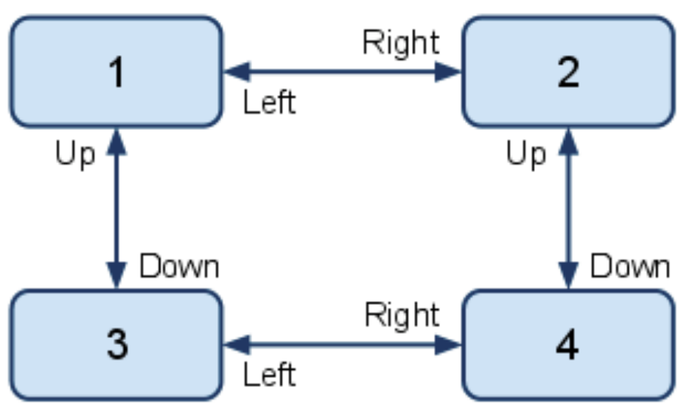
# Practical Constraints of TV

- The expected D-Pad focus model is 2-dimensional
  - The Web platform is used to a linear model

"Traditional Web" model



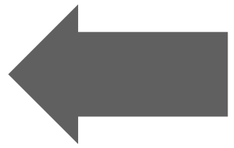

Directional-pad model






#gtvweb

# The Back Key



What happens when the user hits the “Back” key is one of the most-overlooked features of modern web apps.

- Set window.location.hash and respond to hashchange events
- Don’t over-use this, though!

# Technical Points Specific to Google TV

- Autozoom
  - Google TV automatically scales web pages to display properly on TV. If your site attempts to be "pixel-perfect", the algorithm may not be optimal.

  - You can disable auto-zoom:

```
<meta name="gtv-autozoom" content="off" />
```

  - or control it:

```
document.getElementsByTagName('body')[0].style.zoom = 1;
```

# Design Guidance for Web Apps on TV

## Make the most of your space





#gtvweb

# Design Guidance for Web Apps on TV

!=

## Be Sure to Sit Back and Test Your Design on a TV!

# Design Guidance for Web Apps on TV

- Visual feedback and motion
  - Give visual feedback on actions
  - Give directions!
    - Show splash screens,
      have auto-hiding overlay controls
  - Experiences are not “static” on TV
    - Use transitions and animation



```
<style type='text/css'>
  div {
    -webkit-transition-property: opacity, left;
    -webkit-transition-duration: 2s, 4s;
  }
</style>
```

# Tools We Provide for You

#gtvweb

# Google TV UI Templates

- Site templates designed for 10ft living room experience
- Both templates are provided in both HTML5 and Flash, and support D-Pad navigation and video playback controls
- Site templates at goo.gl/guzvn





#gtvweb

# Google TV Web UI Libraries

goo.gl/8ajdi

#gtvweb

# Google TV Web UI Libraries (goo.gl/8ajdi)

Google TV jQuery UI Library

goo.gl/ObULa

```
$('mydiv')
$('.a:even')
$('.a').click(function() { ... })
$('button').html('Look mah')
$("div:hidden:first").fadeIn("slow");
```

Google TV Closure UI Library

goo.gl/sCyj4

```
goog.inherits(childClass, parentClass)
goog.require('tv.ui.Container')
goog.dom.getElement('mydiv')
goog.dom.getElementsByClass('buttons')
goog.debug.expose(person)
```

# Google TV Web UI Libraries (goo.gl/8ajdi)

Google TV jQuery UI Library

goo.gl/ObULa

- Requires more JavaScript
- CSS selectors to specify navigable areas
- UI Control Libraries:
*(Row, SideNav, Sliding, Photo, Video, Roller, Rotator, Stack)*

Google TV Closure UI Library

goo.gl/sCyj4

- Requires more HTML markup
- Special CSS class names to specify navigable areas
- UI building blocks:
*(Grid, scrolling containers, components, buttons, links, input, menu, ... more)*

# Google TV Web UI Libraries (goo.gl/8ajdi)

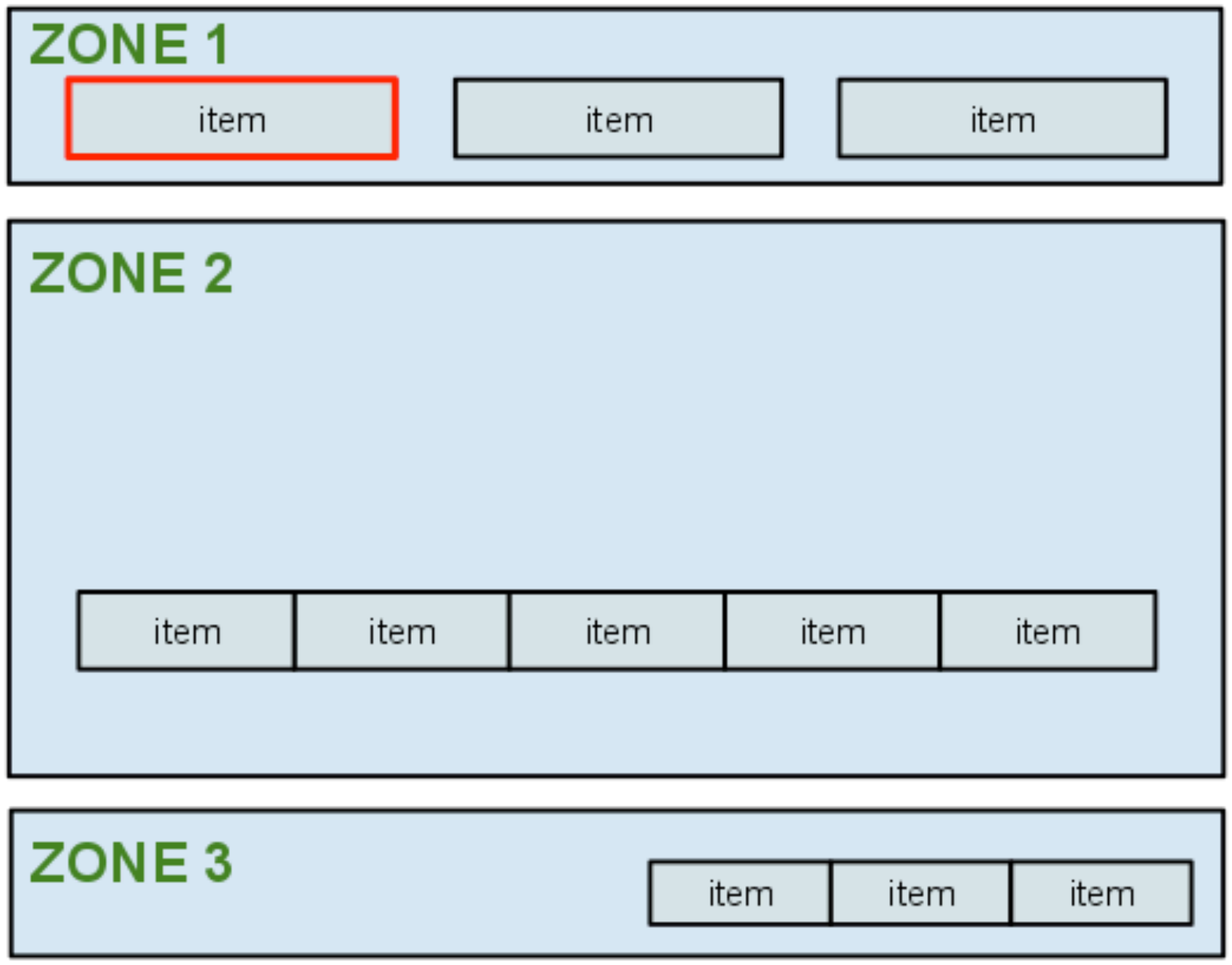
1. HTML: Structure

2. CSS: Presentation

3. JavaScript: Initialize / Events

# Google TV jQuery UI Library (goo.gl/ObULa)

## Keyboard Navigation: Concept




- Key behavior zones (HTML containers)
- Specify zones using CSS selectors:
  `.item:not(.nonav)`
- Remembers last selected item per zone
- *(optional)* Geometry feature:
  Select next closest item based on distance

# Google TV jQuery UI Library (goo.gl/ObULa)

## Keyboard Navigation: HTML

```
<ul id="zone1">
  <div class="item-row">
    <li class="item-parent"><div class="it
    <li class="item-parent"><div class="it
    <li class="item-parent"><div class="it
  </div>
</ul>

<ul id="zone2">
  <div class="item-row">
    <li class="item-parent"><div class="it
    <li class="item-parent"><div class="it
    <li class="item-parent"><div class="it
  </div>
</ul>
```

## Stylize using CSS3

```
ul {
  list-style-type: none;
  margin: 0 auto 10px;
  padding: 10px 20px 15px;
}

.item {
  -webkit-transition: -webkit-box-shadow 300ms ease;
}

#zone1 .item {
  border: 3px solid #89a;
  display: inline-block;
  font-size: 30px;
  margin: 0 15px 0 0;
  padding: 5px 0;
  text-align: center;
  width: 170px;
  -webkit-border-radius: 10px;
}
```

# Google TV jQuery UI Library (goo.gl/ObULa)

## Keyboard Navigation: JavaScript

```
var zone1 = new gtv.jq.KeyBehaviorZone({
  containerSelector: '#zone1',
  navSelectors: {
    item: '.item',
    itemParent: '.item-parent',
    itemRow: '.item-row'
  },
  selectionClasses: {
    basic: 'item-selected'
  },
  keyMapping: {
    13: enterCallback
  }
});
```

```
var zone2 = new gtv.jq.KeyBehaviorZone({
  containerSelector: '#zone2',
  navSelectors: {
    item: '.item',
    itemParent: '.item-parent',
    itemRow: '.item-row'
  },
  selectionClasses: {
    basic: 'item-selected'
  }
});
```

selectors

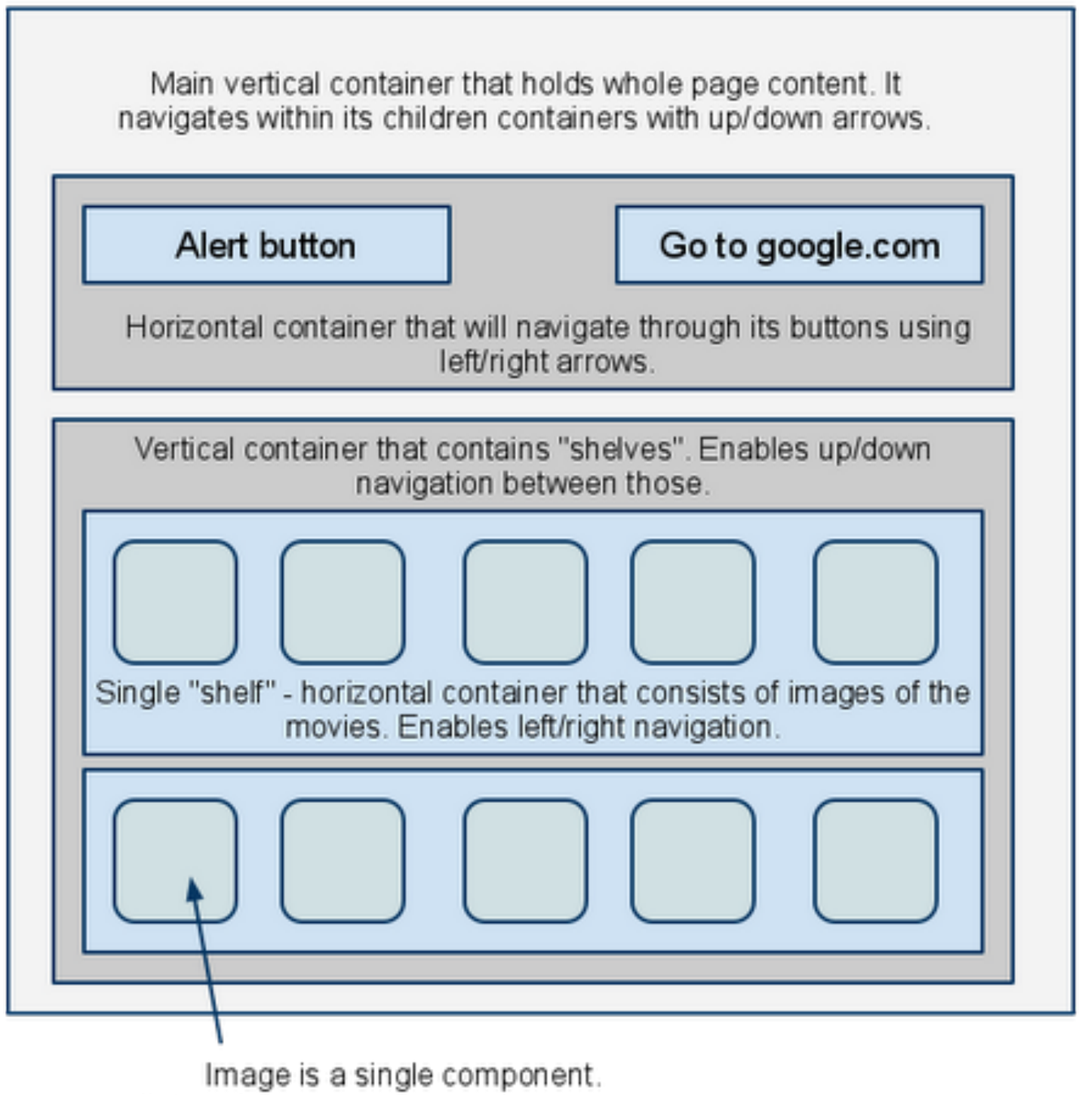
selected

event handler

```
// Add behavior zones
var keyController = new gtv.jq.KeyController();
keyController.addBehaviorZone(zone1);
keyController.addBehaviorZone(zone2);
keyController.start(zone1, true);
```

# Google TV Closure UI Library (goo.gl/sCyj4)

Decorator: Concept



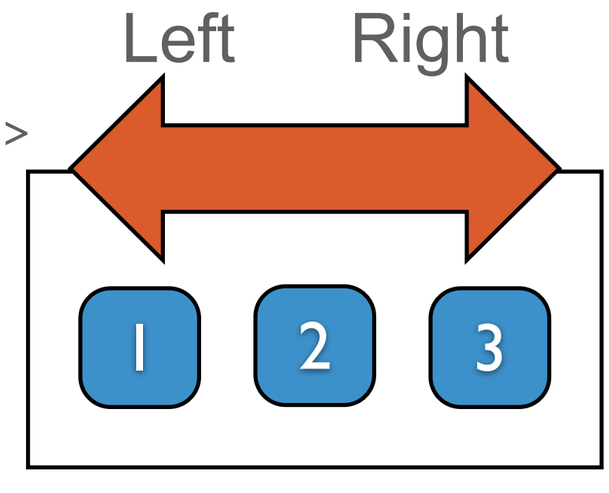

- Horizontal / Vertical container building blocks
- Scrolling containers
- Selectable components, buttons, links, input
- Decorate HTML using semantic class names

```
.tv-container-horizontal
.tv-container-start-scroll
.tv-component
.tv-link
                    ... more
```

# Google TV Closure UI Library (goo.gl/sCyj4)

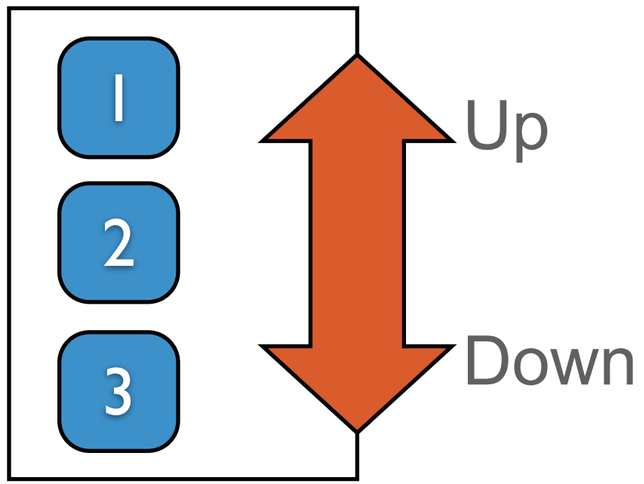
Decorator: HTML

```
<ul class="tv-container-horizontal">
  <li class="tv-component">1</li>
  <li class="tv-component">2</li>
  <li class="tv-component">3</li>
</ul>
```




```
ul {
  list-style-type: none;
  margin: 0 auto 10px;
  padding: 10px 20px 15px;
}
li {
  border: 3px solid #333;
  margin: 0 15px 0 0;
  padding: 10px
  -webkit-border-radius: 10px;
}
.tv-container-horizontal li {
  display: inline-block;
  margin-right: 5px;
}
.tv-container-vertical li {
  display: block;
  margin-bottom: 5px;
}
```

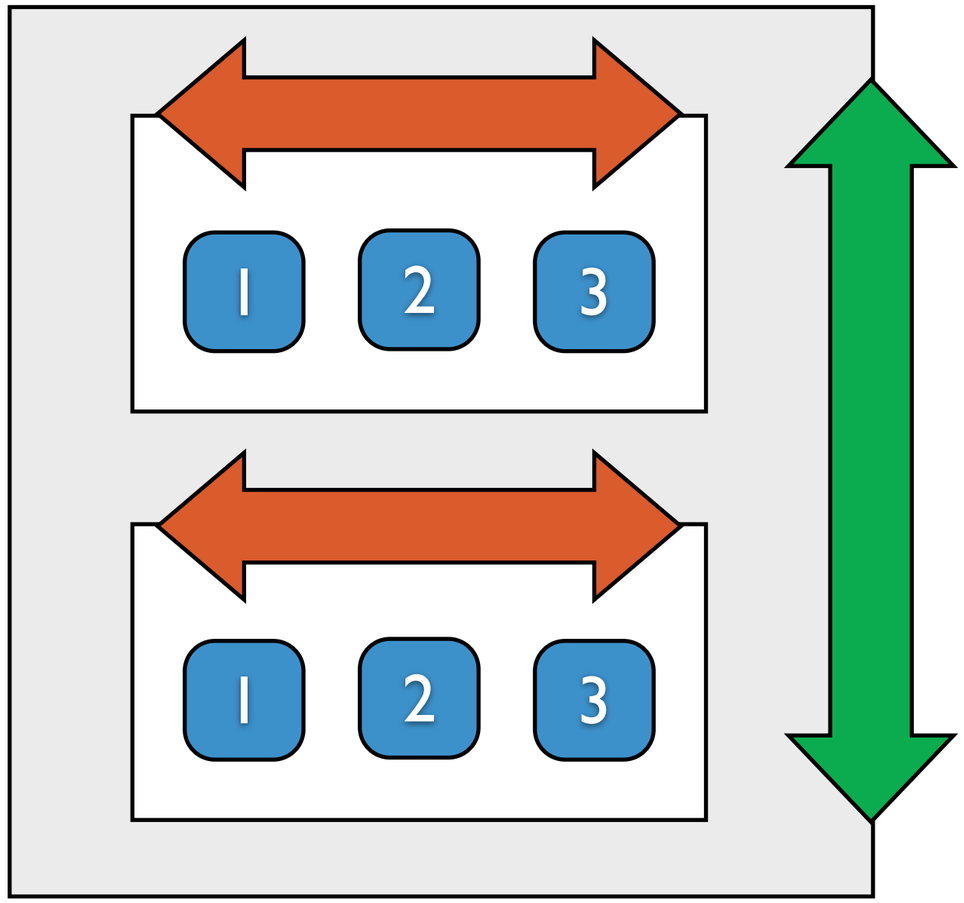
```
<ul class="tv-container-vertical">
  <li class="tv-component">1</li>
  <li class="tv-component">2</li>
  <li class="tv-component">3</li>
</ul>
```



# Google TV Closure UI Library (goo.gl/sCyj4)

Decorator: HTML

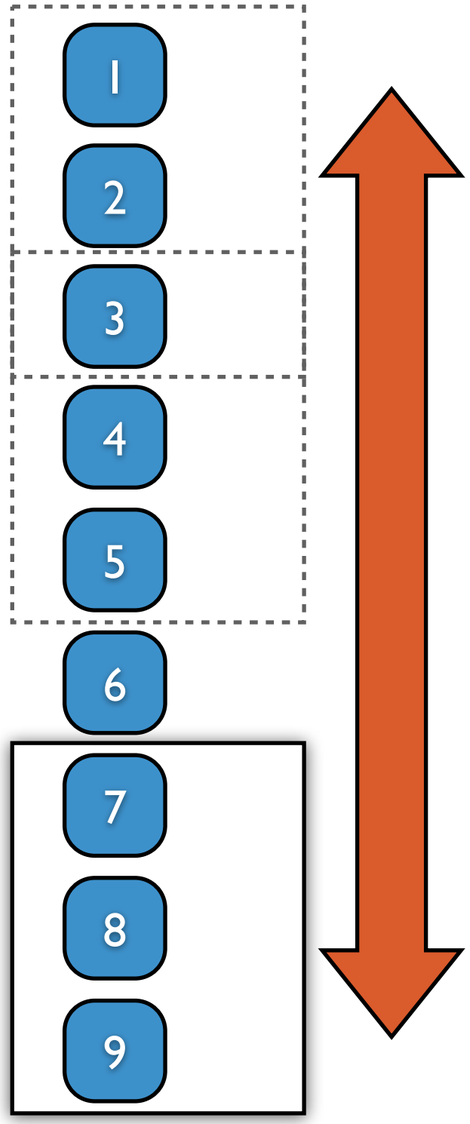
```
<div class="tv-container-vertical">
  <ul class="tv-container-horizontal">
    <li class="tv-component">1</li>
    <li class="tv-component">2</li>
    <li class="tv-component">3</li>
  </ul>

  <ul class="tv-container-horizontal">
    <li class="tv-component">1</li>
    <li class="tv-component">2</li>
    <li class="tv-component">3</li>
  </ul>
</div>
```



# Google TV Closure UI Library (goo.gl/sCyj4)

Decorator: HTML

```
<ul class="tv-container-vertical">
  <div class="tv-container-start-scroll">
    <li class="tv-component">1</li>
    <li class="tv-component">2</li>
    <li class="tv-component">3</li>
    <li class="tv-component">4</li>
    <li class="tv-component">5</li>
    <li class="tv-component">6</li>
    <li class="tv-component">7</li>
    <li class="tv-component">8</li>
    <li class="tv-component">9</li>
  </div>
</ul>
```




```
.tv-container-vertical {
  height: 90px;
  overflow: hidden;
}
```

# Google TV Closure UI Library (goo.gl/sCyj4)

Decorator: JavaScript

```
// Execute the decorator
try {
  tv.ui.decorate(goog.dom.getElement('main'));
} catch (e) {
  alert(e.message);
}

// Set focus on initial element
var focusElement = goog.dom.getElementByClass('first-focus');
var focusComponent = tv.ui.getComponentByElement(focusElement);
focusComponent.tryFocus();
```

# Google TV Web UI Library Demos

goo.gl/ozKzk

#gtvweb

Google I/O 11

# Call to Action

- Think about how your apps apply to the TV space
- Think about new applications you could build
- Think about how these lessons apply to desktop & mobile

- Get a Google TV device and start building!

Code site: goo.gl/FIxrd

Forum: goo.gl/RhuDw

Questions?
Twitter: @googletvdev
Hash Tags: #io2011, #gtvweb

Code site: goo.gl/FIxrd
Forum: goo.gl/RhuDw


FEEDBACK: Please provide feedback
on this session at goo.gl/grOj4