Comp 324/424 - Client-side Web Design

Spring Semester 2024 Week 3

Dr Nick Hayward

HTML Basics - `<body>` - part 6

lists

- unordered list `<ul>` , ordered list `<ol>` , definition list `<dl>`
- `<ul>` and `<ol>` contains list items `<li>`

```
<ul>
  <li>...</li>
</ul>
```

```
<ol>
  <li></li>
</ol>
```

- definition list uses `<dt>` for the item, and `<dd>` for the definition

```
<dl>
  <dt>Game 1</dt>
  <dd>our definition</dd>
</dl>
```

HTML & JS Example - add basic toggle to lists - HTML

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>DOM Manipulation - Node Toggle</title>
  </head>
  <body>
    <header>
      <h3>DOM manipulation - Node Toggle</h3>
    </header>
    <section id="quote">
      <p>
        <blockquote id="berryhead" data-visible="true">
          Shine through the gloom, and point me to the skies...
        </blockquote>
      </p>
    </section>
```

```
    <section id="content">
      <h4>Planets...</h4>
      <div id="list-output">
        <span id="toggle">toggle...</span>
        <ul id="planets" data-visible="true">
          <li>Mercury</li>
          <li>Venus</li>
          <li>Earth</li>
          <li>Mars</li>
          <li>Jupiter</li>
          <li>Saturn</li>
          <li>Uranus</li>
          <li>Neptune</li>
          <li>Pluto</li>
        </ul>
      </div>
    </section>
    <!-- load JS files - pre module design example -->
    <script type="module" src="./toggle.js"></script>
  </body>
</html>
```

HTML & JS Example - add basic toggle to lists - JS Option 1

- various options for toggling visibility of DOM nodes
- option 1 relies on inefficient iterator of child nodes
- nodes may include elements, text, attributes…

```
// toggle switch
let toggle = document.getElementById('toggle');

// get node in DOM
let domNode = document.getElementById('planets');

toggle.addEventListener('click', () => {
  // get child nodes
  let nodeChildren = domNode.children;
  console.log(nodeChildren);
  if (domNode.getAttribute('data-visible') === 'true') {
      domNode.setAttribute('data-visible', 'false');
      // modify display property for each child
      for (let child of nodeChildren) {
        child.style.color = '#779eab';
        child.style.display = 'none';
       }
    } else {
      domNode.setAttribute('data-visible', 'true');
      // modify display property for each child
      for (let child of nodeChildren) {
        child.style.color = '#000';
        child.style.display = 'list-item';
       }
    }
```

```
});
```

- Demo - List Toggle Children

HTML & JS Example - add basic toggle to lists - JS Option 2

- option 2 uses more efficient DOM properties to access required nodes
- access element nodes & ignores text &c. nodes in DOM…

```
// toggle switch
let toggle = document.getElementById('toggle');

toggle.addEventListener('click', () => {
  // get sibling element to toggle...
  let siblingNode = toggle.nextElementSibling;

  // check child node element visibiity
  if (siblingNode.getAttribute('data-visible') === 'true') {
    // update visibility
    siblingNode.setAttribute('data-visible', 'false');
    // hide sibling node
    siblingNode.style.display = 'none';
  } else {
    // update visibility
    siblingNode.setAttribute('data-visible', 'true');
    // show sibling node
    siblingNode.style.display = 'block';
  }
});
```

- Demo - List Toggle Sibling
- JS Info - DOM Nodes

HTML & JS Example - add basic toggle to lists - JS Option 3

- add some initial animation

```
...
function slideUp() {
  if (slideHeight < 1) {
    console.log('slide up - height less than 1...');
    return;
  }
  slideHeight -= 2;
  siblingNode.style.height =  slideHeight + 'px';
  requestAnimationFrame(slideUp);
}

requestAnimationFrame(slideUp);
...
```

- Demo - Toggle vertical with animation
- Demo - Toggle horizontal with animation
- MDN - requestAnimationFrame

HTML Basics - `<body>` - part 7

forms

- used to capture data input by a user, which can then be processed by the server
- `<form>` element acts as the parent wrapper for a form
- `<input>` element for user input includes options using the *type* attribute
  - text, password, radio, checkbox, submit

```
<form>
  Text field: <input type="text" name="textfield" />
</form>
```

- process forms using
  - e.g. JavaScript…
- MDN - HTML Forms

Semantic HTML - intro

- importance of web standards
  - and their application to HTML markup and documents
- standards help drive a consideration of markup, e.g. HTML
  - usage for what they mean
  - not simply how they will look…
- semantic instead of purely presentational perspective
  - introduction of meaning and value to the document
- when pages are processed
  - impart structure and meaning beyond mere presentation
- a core consideration for usage of markup languages
- issues persist with HTML element usage
  - e.g. inline elements such as `<b>` and `<i>`

Semantic HTML - a reason to care

- Semantic HTML - opportunity to convey meaning with your markup
  - meaning may be explicit due to the containing element
  - implicit due to a structured grouping of elements
- markup makes it explicit to the browser
  - underlying meaning of a page and its content
- notion of meaning and clarity also conveyed to search engines
  - fidelity with query and result…
- semantic elements provide information beyond page rendering and design
- use semantic markup correctly
  - create more specific references for styling
  - greater chance of rendering information correctly

Semantic HTML - example usage

```
<!-- incorrect element chosen -->
<div id="code">
document.addEventListener('click', function () {
  console.log('Click received...');
});
</div>
```

```
<!-- correct element chosen -->
<code>
document.addEventListener('click', function () {
  console.log('Click received...');
});
</code>
```

- Demo - semantic example usage

Semantic HTML - correct usage

- need to ensure elements convey their correct meaning
  - i.e. the meaning expected for the contained content
- e.g. often see the following elements mis-used and applied incorrectly for markup,
  - `<p>` - paragraphs
  - `<ul>` - unordered list
  - `<h1>` to `<h6>` - headings
  - `<blockquote>` - blockquote
- using `<blockquote>` to simply help indent text
  - instead of CSS margins…
- or the perennial mis-use of a `<p>`
  - simply add extra space between elements

```
<p> </p>
```

HTML5 - intro

- finally became a standard in October 2014
- introduces many new features to HTML standard
- additional features include, e.g.
  - new canvas element for drawing
  - video and audio support
  - support for local offline storage
  - content specific elements
    - including article, footer, header, nav, section
  - form controls such as
    - calendar, date, time, email, url, search
- new input type attribute values
  - assigned to provide better input control
- Check browser compatibility using HTML5 Test

HTML5 - basic template

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>

  </body>
</html>
```

HTML5 - Elements - part 1

- often known simply as **tags**
- elements allow us to add a form of metadata to our HTML page
- for example, we might add

```
<!-- a paragraph element -->
<p>add some paragraph content...</p>
<!-- a first heading element -->
<h1>our first heading</h1>
```

- this metadata used to apply structure to a page's content

HTML5 - Elements - part 2

- we can now add additional structure to our basic template

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- title for the web page appears in the window, tab heading... -->
    <title>Demo 1</title>
  </head>
  <body>
    <h1>Our first web page</h1>
    <p>
      As we build our web apps, more elements and content will be added...
    </p>
  </body>
</html>
```

- Demo - Our first web page

HTML5 - Comments

- comments are simple and easy to add to HTML
- add to HTML code as follows,

```
<!-- a comment in html -->
```

- comment not explicitly visible to the user in the rendered page
- comment appears in the code for reference...

Image - HTML5 sample rendering 1

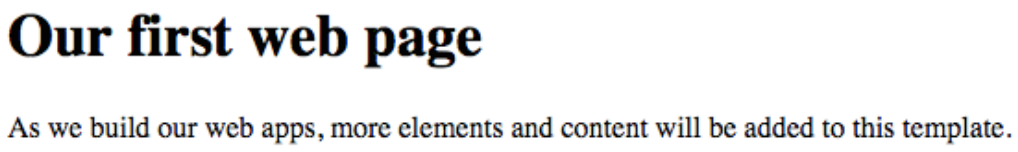

Figure 1: HTML - sample rendering of demo 1

Source - Demo 1

HTML5 - semantic elements - part 1

- new semantic elements added for HTML5
- known as **block-level** elements
  - includes the following elements,

```
<article>
<aside>
<details>
<figure>
<figcaption>
<footer>
<header>
<main>
<nav>
<section>
```

- better structure underlying documents
  - add clear semantic divisions

HTML5 - semantic elements - part 2

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <!-- our second demo with lots of new elements -->
    <title>Demo 2</title>
  </head>
  <body>
```

```
    <header>
      <h1>Our first web page</h1>
    </header>
    <!-- primary navigation elements, links... -->
    <nav>Option 1</nav>
    <!-- main content -->
    <main>
      <section>
        <p>
          As we build our web apps, more elements and content will be added...
        </p>
        <figure>
          <img src="media/images/philae-demo2.jpg" alt="temple of philae" width="333px"
                        height="200px">
        </figure>
      </section>
      <aside>
        Temple at Philae in Egypt is Ptolemaic era of Egyptian history...
      </aside>
    </main>
    <footer>
      foot of the page...
    </footer>
  </body>
</html>
```

- Demo - New elements added

Image - HTML5 sample rendering 2

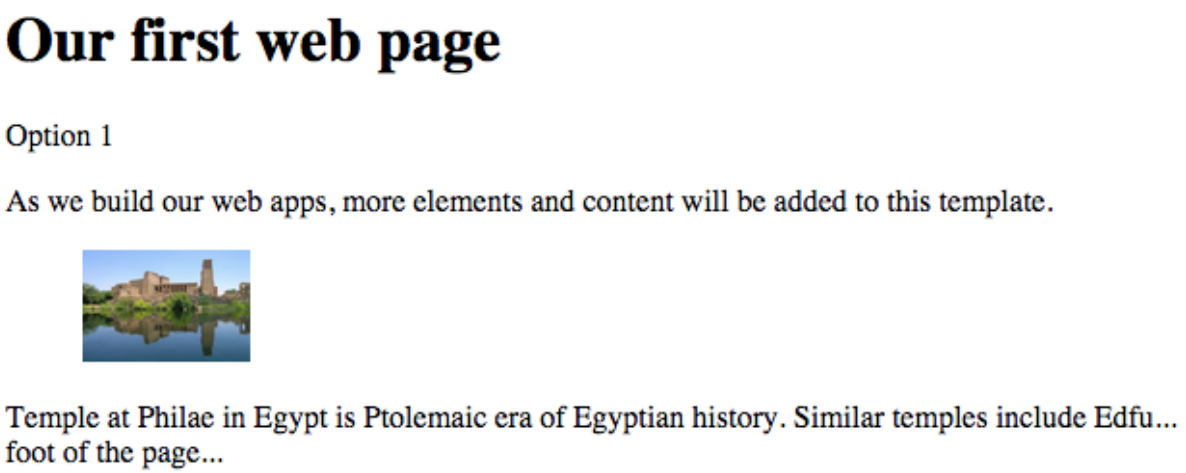

Figure 2: HTML - sample rendering of demo 2

Source - Demo - New elements added

HTML5 - semantic elements - part 3

- element tag `article` not used in previous demo
- `article` and `section` tag can both cause some confusion
- not as widely used as expected
- `div` element still widely seen in development
- HTML5 is supposed to have relegated `div`
  - sectioning element of last resort…
- `article` and `section`
  - good analogy with a standard newspaper
  - different sections such as headlines, politics, health…
  - each section will also contain articles
- HTML specification also states that an `article` element

  > represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication.

HTML5 - semantic elements and structure - intro

- perceived issue or concern with HTML5 semantic elements
  - how and when to add them to our document
  - where and when do we add them to our page?
- non-semantic elements often considered simpler to apply
  - generalised application and context for usage

HTML5 - semantic elements and structure

header and nav

- `<header>`
  - used to collect and contain introductory content
  - semantically appropriate for the head or top of a page
  - technically feasible and acceptable to include multiple `<header>` elements
    - e.g. `<header>` within main content, sidebar content, an article, a section…
- `<nav>`
  - short for *navigation*
  - stores and defines a set of links for internal or external navigation
  - not meant to define all page navigation links
  - often considered suitable for primary site links
  - additional links can be placed in
    - sidebar, footer, main content…
    - no need to consider a `<nav>` element for these links…

HTML5 - Semantic elements and structure

main

- this element tag defines our **main** content
- traditionally the central content area of our page or document
- HTML4 often used a `<div>` element

  - plus a `class` or `id` to define central content
  - e.g.

```
<!-- e.g. HTML4 main content -->
<div id="main">
  ...
</div>
```

- HTML5 semantically defines and marks content as `<main>`
- `<main>` should not include any page features such as
  - nav links, headers &c., that are repeated across multiple pages
- cannot add multiple `<main>` elements to a single page
- must not be structured as a child element to
  - `<article>`, `<aside>`, `<footer>`, `<header>`, or `<nav>`

HTML5 - Semantic elements and structure

section, article, aside - part 1

- `<section>`
  - defines a section of a page or document
  - W3C Documentation defines as follows,

    > a section is a thematic grouping of content. The theme of each section should be identified, typically by including a heading as a child of the section element.

- a site can be sub-divided into multiple `<section>` groupings
  - e.g. as we might consider a chapter or section break in a book…
- `<article>`
  - suitable for organising and containing independent content
  - include multiple `<article>` elements within a page
  - use to establish logical, individual groups of content
    - again, newspaper analogy is useful to remember
    - e.g. a blog post, story, news report…might be a useful article
  - key to using this element is often whether content can be used in isolation
- `<aside>`
  - used to define some content aside from containing parent content
  - normally used to help define or relate material to surrounding content
  - effectively acts as supporting, contextual material

HTML5 - Semantic elements and structure

section, article, aside - part 2

- MDN Documentation suggests,

  > if it makes sense to separately syndicate the content of a `<section>` element, use an `<article>` element instead

and

> do not use the `<section>` element as a generic container; this is what `<div>` is for, especially when the sectioning is only for styling purposes. A rule of thumb is that a section should logically appear in the outline of a document.

HTML5 - Semantic elements and structure

figure, figcaption

- `<figure>` & `<figcaption>`
  - as with print media, we can logically group image and caption
  - `<figure>` acts as parent for image grouping
  - child elements include
    - `<img>` and `<figcaption>`

```
<figure>
<img src="media/images/philae-demo2.jpg" alt="temple of philae"
width="333px" height="200px">
<figcaption>Ptolemaic temple at Philae, Egypt</figcaption>
</figure>
```

- updated demo with figure grouping
  - Demo - Semantic structuring

HTML5 - Semantic elements and structure

footer

- `<footer>`
  - usually contains information about its containing element
- example 1 - in a footer for an article
  - might use this element to define and record
    - author of the article
    - publication date
    - suitable tags or metadata
    - associated documents…
- example 2 - a footer simply placed at the **foot** of a page
  - record copyright information
  - contextual links
  - contact information
  - small logos…
- example 2 considered standard usage for `<footer>`
  - continues from HTML4 and earlier generic usage…

Image - HTML5 page structure - part 1

semantic elements

Image - HTML5 page structure - part 2

semantic elements

HTML5 page structure - part 3

- not included `<html>` and `<body>` tags in diagrams
  - required for all HTML documents

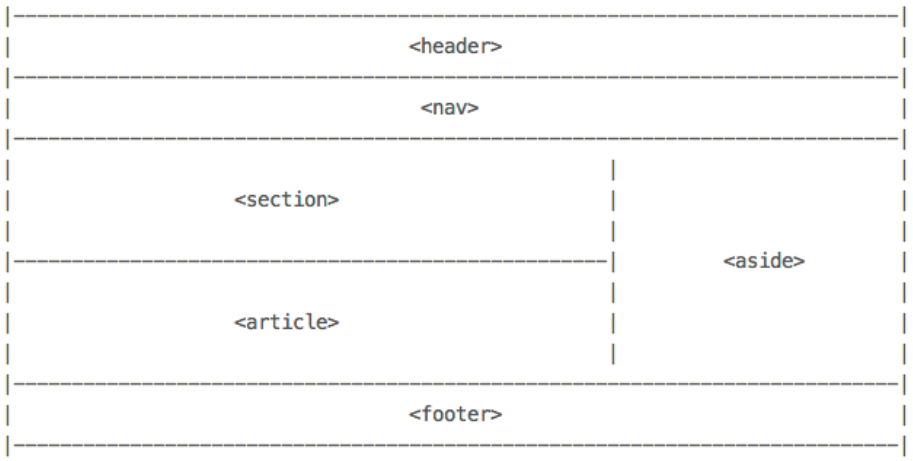

Figure 3: HTML5 - Structure

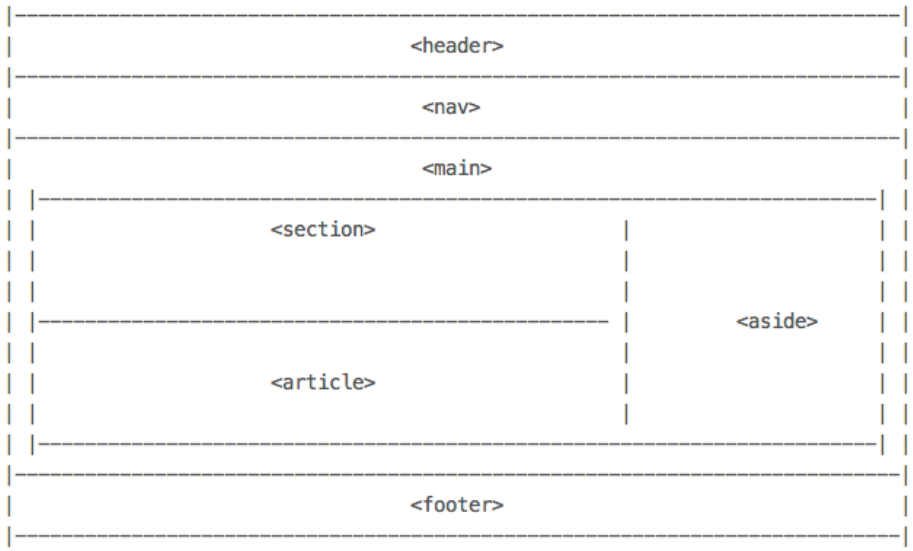

Figure 4: HTML5 - Structure

- divided the page into four logical, semantic divisions
  - header
  - nav
  - main
  - footer
- we could also add a sidebar &c. for further division of content

---

Design considerations - poor design

Design consideration - good design

Design considerations - relevance of design

HTML5 - extra elements

intro

- many other interesting and useful new HTML5 elements
  - in addition to semantic elements
- some struggle for browser compatibility
- useful new elements such as
  - graphics and media
- HTML5 APIs introduced as well, including

Figure 5: 1931 London Underground Map

Figure 6: No Camping!

  - App Cache
  - Drag/Drop
  - Geolocation
  - Local Storage
  - …
- again, check browser support and compatibility
- "Can I Use" browser check
- Can I Use?
  - e.g. Can I Use Drag and Drop?

HTML5 - Extra elements - media - part 1

video `<video>` element

- until HTML5, video playback reliant on plugins
  - e.g. Adobe Flash
- embed video using element tag `<video>`
- add attributes for
  - height, width, controls…
- not all web browsers support all video codecs
- option to specify multiple video sources
- best supported codecs include
  - MP4 (or H.264), WebM, OGG…
- good general support for `<video>` element

- check browser support for `<video>` element
  - Can I use - video?

HTML5 - Extra elements - media - part 2

video example `<video>` - a quick example might be as follows,

```
<video width="300" height="240" controls>
  <source src="media/video/movie.mp4" type="video/mp4">
  <source src="media/video/movie.webm" type="video/webm">
  Your browser does not support the video tag.
</video>
```

- Demo - HTML5 Video playback

HTML5 - Extra elements - media - part 3

audio `<audio>` element

- HTML5 also supports standardised element for embedded audio
- supported codecs for `<audio>` playback include
  - MP3 and mp4
  - WAV
  - OGG Vorbis
  - 3GP
  - m4a
- again, check browser support and compatibility
  - Can I use - audio?
- fun test of codecs
  - HTML5 Audio

HTML5 - Extra elements - media - part 4

audio example `<audio>` - a quick example might be as follows,

```
<audio controls>
  <source src="media/audio/audio.mp3" type="audio/mpeg">
  Your browser does not support the audio tag.
</audio>
```

- Demo - HTML5 Audio playback

HTML5 - Extra elements - graphics - part 1

canvas

- graphics elements are particularly fun to use
- use them to create interesting, useful graphics renderings
- in effect, we can draw on the page
- `<canvas>` element acts as a placeholder for graphics
  - allows us to draw with JavaScript
- draw lines, circles, text, add gradients…

- e.g. draw a rectangle on the canvas

HTML5 - Extra elements - graphics - part 2

canvas example `<canvas>` will be created as follows,

```
<canvas id="canvas1" width="200" height="100">
  Your browser does not support the canvas element.
</canvas>
```

then use JavaScript to add a drawing to the canvas

```
<script type="text/javascript">
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,150,75);
</script>
```

Result is a rendered black rectangle on our web page.

- Demo - HTML5 Canvas - Rectangle

HTML5 - Extra elements - graphics - part 3

canvas example A square can be created as follows,

```
<script type="text/javascript">
function draw() {
/*black square*/
var can1 = document.getElementById("canvas1");
var context1 = can1.getContext("2d");
context1.fillStyle="#000000";
context1.fillRect(0,0,50,50);
}
</script>
```

Again, we end up with the following rendered shape on our canvas.

- Demo - HTML5 Canvas - Square

HTML5 - Extra elements - graphics - part 4

canvas examples

- modify drawing for many different shapes and patterns
  - simple lines, circles, gradients, images…

1. shows different rendered shapes on a canvas.

- Demo - HTML5 Canvas - Assorted Shapes

2. little retro games

- Demo - HTML5 Canvas - Retro Breakout Game

HTML5 - Extra elements - graphics - part 5

canvas examples - basics

- basic drawing - rectangle & staircase
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic/
- Example - basic drawing - stepped pyramid
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic2/
- Example - various colours
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic3/
- Example - basic drawing - rectangle outlines
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic4/
- Example - draw lines - line & pyramid
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic5/
- Example - draw a stickman
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic6/
- Example - fill paths
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic7/

HTML5 - Extra elements - graphics - part 6

canvas examples - curves & circles

- Example - arcs and circles
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic8/
- Example - Bézier curves - quadratic
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-quadratic/
- Example - Bézier curves - cubic
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic9-cubic/
- Example - arcs and circles - combine shapes to create an *ankh*
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic10-ankh/
- Example - circle function
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic11-function-circles/

HTML5 - Extra elements - graphics - part 7

canvas examples - animation & fun

- Example - horizontal animation
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation1/
- Example - animate size
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation2/
- Example - variant mouse colours
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.1/
- Example - variant mouse colours
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.2/
- Example - random movement and animation
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-animation/animation3.3/

HTML5 - Extra elements - graphics - part 8

canvas examples - images & files

- Example - draw image to canvas from local file
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic1/
- Example - draw image to canvas from local file - `dw` & `dh`
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic2/
- Example - draw image to canvas from local file - `dw` & `dh` plus source crop
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-image/basic3/

HTML5 - Extra elements - graphics - part 9

canvas examples - move & control

- Example - move ball with keyboard control
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move1/
- Example - update `move()` to check canvas boundaries
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move2/
- Example - move ball on 4-point axis
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move3/
- Example - move sprite image
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-sprite-move1/
- Example - move sprite image
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move4/
- Example - check basic collision against blocks
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move5/
- Example - check basic collision against blocks - horizontal
  - http://linode4.cs.luc.edu/teaching/cs/demos/125/drawing/basic-game/basic-ball-move6/

HTML5 - Extra elements - graphics - part 10

more fun canvas examples Some fun examples of animations with HTML5 Canvas API.

- Destroy things in a video - http://www.craftymind.com/factory/html5video/CanvasVideo.html
- Particles - https://codepen.io/eltonkamami/pen/ECrKd
- Curtain - https://codepen.io/dissimulate/pen/KrAwx
- Jelly - https://codepen.io/dissimulate/pen/dJgMaO
- Canvas cycle - http://www.effectgames.com/demos/canvascycle/

Vision & Interfaces

text example 1 Test 1

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc et libero et mi porttitor scelerisque. Mauris gravida enim nec mi vulputate, quis aliquet dolor suscipit. Aenean rutrum sapien vitae lobortis bibendum. Donec vitae interdum diam. Maecenas dapibus facilisis elit vel imperdiet. Cras ultrices tempor dictum. Fusce ex eros, egestas at congue non, venenatis nec nisl. Donec fringilla pulvinar augue eu vulputate. Etiam metus est, aliquam quis sem et, ultricies tincidunt arcu. Integer eu sem nisi. Proin gravida odio urna, vitae scelerisque enim ornare et. Integer placerat massa viverra, aliquam arcu et, porta augue. Aliquam erat volutpat.

Vision & Interfaces

text example 2 Test 2

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc et libero et mi porttitor scelerisque. Mauris gra**v**ida enim nec mi **v**ulputate, quis aliquet dolor suscipit. Aenean rutrum sapien **v**itae lobortis bibendum. Donec **v**itae interdum diam. Maecenas dapibus facilisis elit **v**el imperdiet. Cras ultrices tempor dictum. Fusce ex eros, egestas at congue non, **v**enenatis nec nisl. Donec fringilla pul**v**inar augue eu **v**ulputate. Etiam metus est, aliquam quis sem et, ultricies tincidunt arcu. Integer eu sem nisi. Proin gra**v**ida odio urna, **v**itae scelerisque enim ornare et. Integer placerat massa **vi**verra, aliquam arcu et, porta augue. Aliquam erat **v**olutpat.

Vision & Interfaces

text example 3 Test 3

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc et libero et mi porttitor scelerisque. Mauris gravida enim nec mi vulputate, quis aliquet dolor suscipit. Aenean rutrum sapien vitae lobortis bibendum. Donec vitae interdum diam. Maecenas dapibus facilisis elit vel imperdiet. Cras ultrices tempor dictum. Fusce ex eros, egestas at congue non, venenatis nec nisl. Donec fringilla pulvinar augue eu vulputate. Etiam metus est, aliquam quis sem et, ultricies tincidunt arcu. Integer eu sem nisi. Proin gravida odio urna, vitae scelerisque enim ornare et. Integer placerat massa viverra, aliquam arcu et, porta augue. Aliquam erat volutpat.

Video - Vision & Interfaces

UI Pop Samsung One UI 2: Designed for everyday simplicity

Source - One UI 2 - YouTube

Image - Vision & Interfaces

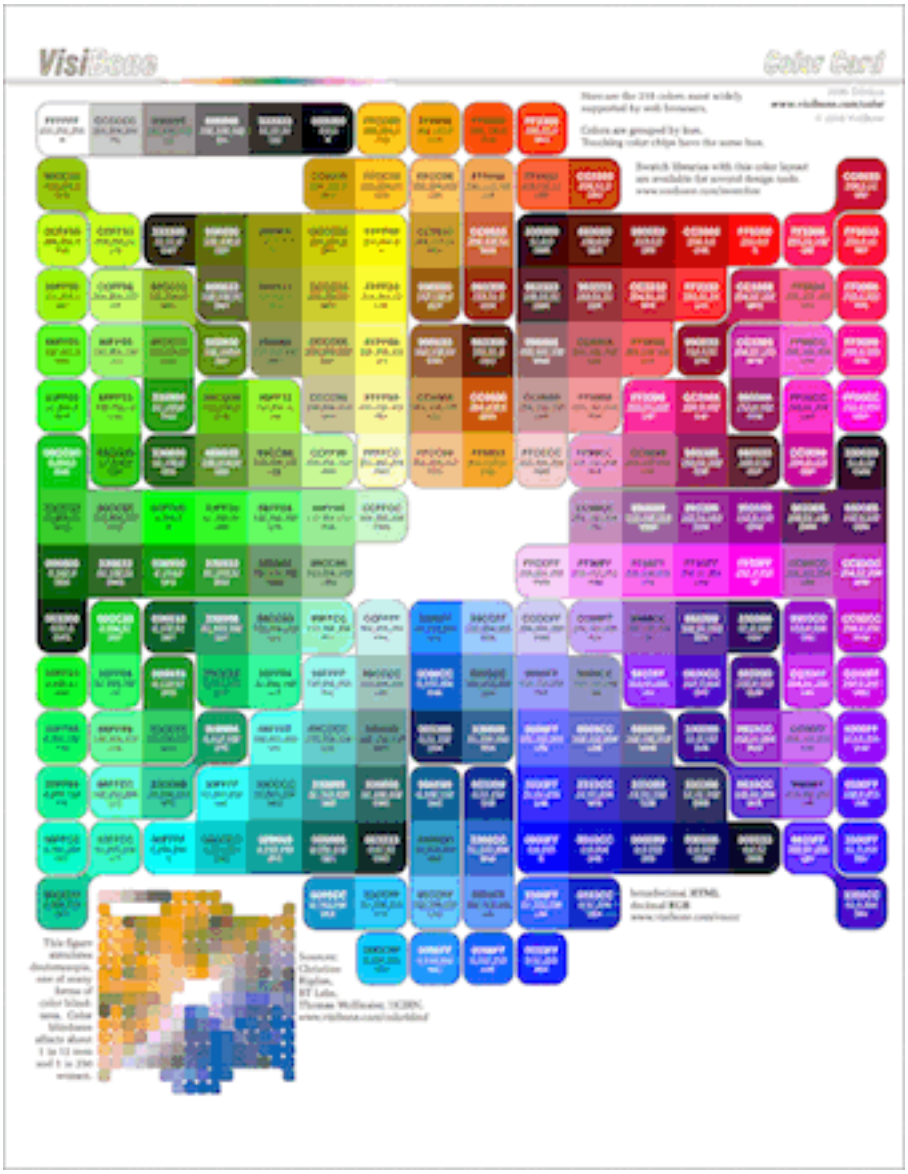

Figure 7: Safe Colours

web safe & browser colours Browser colours & colour blindness (source: VisiBone)

Image - Vision & Interfaces

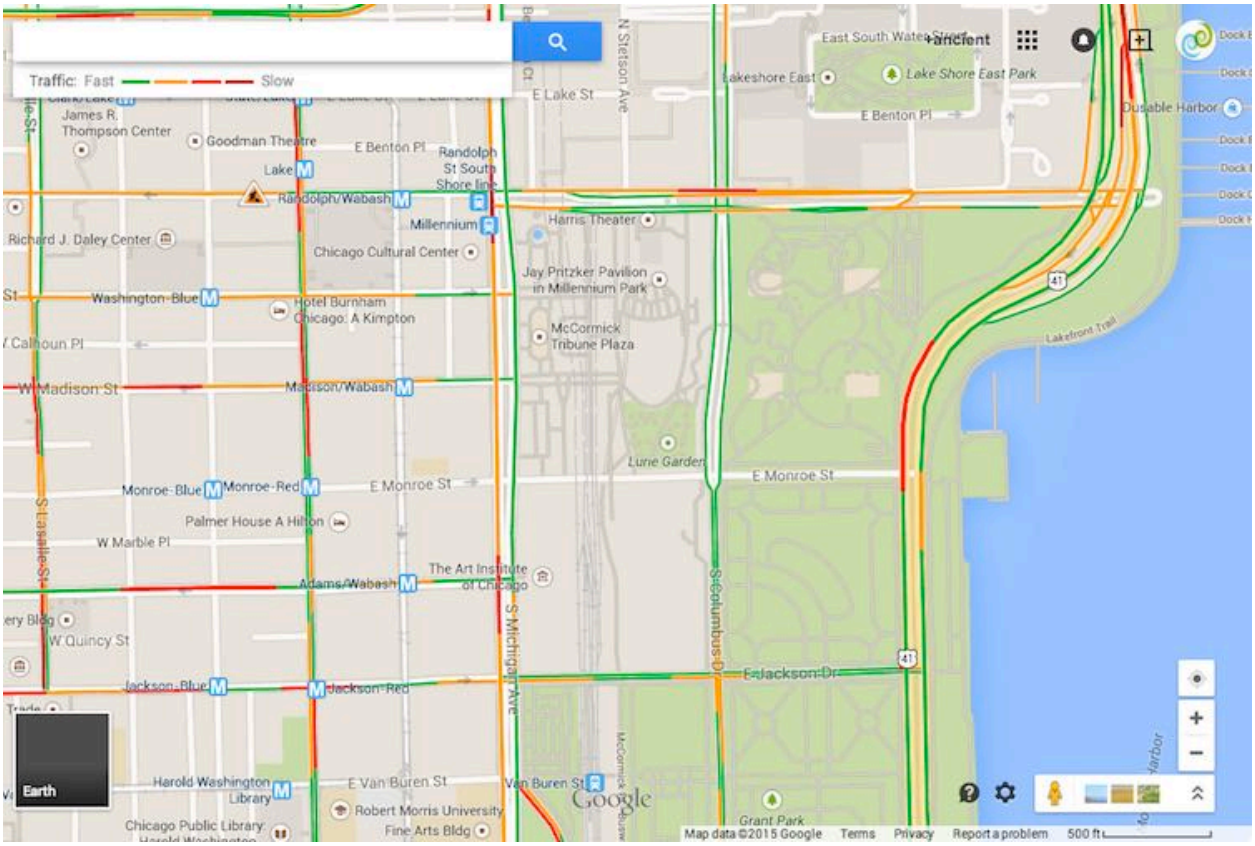

Figure 8: Traffic

design pop...

Demos

- Basic group
- Basic group style
- Basic list items
- Basic menu tabs
- Basic table caption
- Basic table for presentation
- Basic table with accessibility
- Basic table with head, foot, body
- Basic table with headers
- Basic table with summary
- List Toggle Children
- List Toggle Sibling
- Semantic example usage
- Toggle vertical with animation
- Toggle horizontal with animation

HTML5

- Basic structure
- New elements added
- Semantic structuring

Resources

- HTML Colour Picker
- MDN Documentation
  - Block-level Elements
  - Content Categories
  - HTML developer guide
  - Section element
- W3C Documentation
  - Section element