The Four Phases of Web Development

Stages 3 and 4: Develop & Deploy

Domenic J. Licata, Instructional Support Technician
University at Buffalo Department of Art
djlicata@buffalo.edu
4 phases of web development
4 phases of web development

1. Discovery
4 phases of web development

1. Discovery

2. Design
4 phases of web development

1. Discovery
2. Design
3. Develop
4 phases of web development

1. Discovery
2. Design
3. Develop
4. Deploy
Phase 4: develop

1. Write clean, valid code: HTML, CSS, WP Theme
2. Optimize images and video
   - Smallest acceptable file size
   - Use pure CSS in place of images where possible
3. Testing
   - Chrome Dev Tools: Network; Audits
   - Cross browser/device testing
   - Validate: W3C Validator http://validator.w3.org/
Convert Sketches to HTML and CSS

• Sketch out your wireframes - Simplify!

• Compute grid dimensions in pixels.

• Convert static pixel grid to fluid percentages.

• Build out wireframe in HTML and CSS, starting with the largest, most general areas, like #wrapper and #main, down to the smallest details, like buttons.

• Discover breakpoints and add media queries.
Plan For Responsiveness

- Inspect the Robot or Not website (from Ethan Marcotte’s book, Responsive Web Design.)

- This is a complex site, with lots going on. You will not be building sites this complex, but you may find helpful tips by looking through the code.
Building From Wireframes to Code
Building From
Wireframes to Code

• Carefully compute and markup the measurements of each component of your sketches/wireframes.
Building From Wireframes to Code

• Carefully compute and markup the measurements of each component of your sketches/wireframes.

• Which components can be grouped together within HTML5 semantic elements or generic DIVs?
Building From Wireframes to Code

• Carefully compute and markup the measurements of each component of your sketches/wireframes.

• Which components can be grouped together within HTML5 semantic elements or generic DIVs?

• Consider how each element will flow, float and clear in the overall HTML:
  
  • By default, browsers place a line break after each <div> element. Overridden with floats (moves a <div> left or right) and clears (forces a line break left and right)
Building From Wireframes to Code

- Carefully compute and markup the measurements of each component of your sketches/wireframes.

- Which components can be grouped together within HTML5 semantic elements or generic DIVs?

- Consider how each element will flow, float and clear in the overall HTML:
  - By default, browsers place a line break after each `<div>` element. Overridden with floats (moves a `<div>` left or right) and clears (forces a line break left and right)

- Create your elements in your HTML according to your flow analysis, and give them width, height (if necessary) and position in CSS.
HTML
HTML Tags

- **HTML** is plain text consisting of tags which define the content of a page. Tags define the elements which make up the structure of the page and the content.

- **Structural tags**, like `<h1>` through `<h6>` headings and `<p>` paragraphs define the hierarchical flow of elements on a page. Hierarchy refers to a logical progression of the importance of items on a page – `<h1>` is the most important headline; `<h2>` second most important; and so on.

- **Semantic tags** define extra info, like where to place `<em>` emphasis in a sentence.
Structure, not Style

• tags must not be used to define the look of content (style.) Instead, tags define the semantic structure of a page, outlining the relative importance of each chunk of information being presented
Block Level vs Inline

- **Block Level Elements**
  - Stack on top of each other
  - Will start on a new line and cause a new line to be created at the close.
    - `<h1>` through `<h6>` Head 1 through Head 6
    - `<p>` paragraph
    - `<ul>` unordered list (bullets)
Block Level vs Inline

• Inline Elements
  
  • Will appear next to each other as long as there is room within the parent container.
  
  • Do not cause a line break.

  • <em>emphasis, or implied importance (looks like italic)

  • <i>italic - looks like emphasis, but semantically implies alternate voice or mood.

  • <strong>implies extra importance (looks like bold)

  • <span>range of selected text, commonly used to apply a define a class to which a style can be applied

  • <a>anchor (links)
HTML5 Semantic Elements

- header
- main
- section
- article
- nav
- aside
- footer
HTML5 Element Flowchart
Sectioning content elements and friends

By @riddle & @boblet
www.html5doctor.com

Start
A block of flow content (not inline phrasing content)

Is it a major navigation block? Yes
A block of flow content that is intended to be used for navigation
 evacuation purposes. Usually this will be a <nav> element.

Is it required to understand the current content? No
A block of flow content that will not be missed if removed from the page. Usually this will be a <main> element.

Is it required to understand the current content? Yes
A block of flow content that is required to understand the current content. Usually this will be a <article> element.

Could you move it to an appendix? No
A block of flow content that is not required to understand the current content. Usually this will be a <section> element.

Could you move it to an appendix? Yes
A block of flow content that will be missed if removed from the page. Usually this will be a <aside> element.

Is it logical to add a heading? No
A block of flow content that does not require a heading. Usually this will be a <figure> element.

Is it logical to add a heading? Yes
A block of flow content that requires a heading. Usually this will be a <section> element.

Is it required to understand the current content? No
A block of flow content that will not be missed if removed from the page. Usually this will be a <main> element.

Is it required to understand the current content? Yes
A block of flow content that is required to understand the current content. Usually this will be a <article> element.

Could you move it to an appendix? No
A block of flow content that is not required to understand the current content. Usually this will be a <section> element.

Could you move it to an appendix? Yes
A block of flow content that will be missed if removed from the page. Usually this will be a <aside> element.

Is it logical to add a heading? No
A block of flow content that does not require a heading. Usually this will be a <figure> element.

Is it logical to add a heading? Yes
A block of flow content that requires a heading. Usually this will be a <section> element.

Appropriate element
Flow content with no additional semantics, e.g. for CSS hooks...

*Sectioning content element
These four elements (and their headings) are used by HTML5's outline algorithm to make the document's outline.

html5doctor.com/outline

html5doctor.com/semantics

html5doctor.com/article

html5doctor.com/aside

html5doctor.com/figure

html5doctor.com/section

html5doctor.com/div

2011-07-23 v1.5
For more information:
Hyperlinks (<a> anchor tags) are the heart of HTML. Clicking a link sends a request to a web server to send a different page (or a specific location of a page) to the browser.

A link (URL) is the address (or the path) to the location of the new page.

A link can call for a page located on the same website or on a different site altogether.
An **absolute path** points to a specific page on a specific server using a full URL

<http://art.buffalo.edu/exercises/ex01/instructions.html>

A **document-relative path** points to a file relative to the file you are currently browsing

<../exercises/ex01/instructions.html> (one level up out of the current directory and into the folder named “exercises”)

A **root-relative path** describes the location of a file relative to the top level of a site

</exercises/ex01/instructions.html> (at the top level of the website in the exercises folder.)
URLs & `<a>` tags

### Uniform Resource Locators

#### Fully Resolved URL

http://www.art.buffalo.edu/undergraduate/programs.html

#### Absolute Path

```html
<a href = "http://www.art.buffalo.edu/undergraduate/programs.html">VS</a>
```

#### Document Relative Path

```html
<a href = "../undergraduate/programs.html">Undergrad Programs</a>
```

#### Root Relative Path

```html
<a href = "/degrees/undergraduate/programs.html">Undergrad Programs</a>
```

### Domain Name Servers
directories & paths

From HTML & CSS: design and build websites
### Relative Link Type

<table>
<thead>
<tr>
<th>Relative Link Type</th>
<th>Example (from diagram on previous page)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SAME FOLDER</strong></td>
<td>To link to a file in the same folder, just use the file name. (Nothing else is needed.)</td>
</tr>
<tr>
<td><strong>CHILD FOLDER</strong></td>
<td>To link to music reviews from the music homepage:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;a href=&quot;reviews.html&quot;&gt;Reviews&lt;/a&gt;</code></td>
</tr>
<tr>
<td><strong>GRANDCHILD FOLDER</strong></td>
<td>To link to music listings from the homepage:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;a href=&quot;music/listings.html&quot;&gt;Listings&lt;/a&gt;</code></td>
</tr>
<tr>
<td></td>
<td>To link to DVD reviews from the homepage:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;a href=&quot;movies/dvd/reviews.html&quot;&gt;Reviews&lt;/a&gt;</code></td>
</tr>
<tr>
<td><strong>PARENT FOLDER</strong></td>
<td>To link to the homepage from the music reviews:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;a href=&quot;/../index.html&quot;&gt;Home&lt;/a&gt;</code></td>
</tr>
<tr>
<td><strong>GRANDPARENT FOLDER</strong></td>
<td>To link to the homepage from the DVD reviews:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;a href=&quot;/../index.html&quot;&gt;Home&lt;/a&gt;</code></td>
</tr>
</tbody>
</table>
<a href="mailto:joejones@example.org">Email Joe</a>
opening links in a new window*

<a href="http://eff.org" target="_blank">EFF</a>
(opens in a new window)

EFF (opens in a new window)

* frowned upon
Linking to parts of the same page or to parts of a different page

Modified from HTML & CSS: design and build websites
Linking to parts of the same page or to parts of a different page

Film-Making Terms

- Arc Shot
- Interlude
- Prologue

Arc Shot
A shot in which the subject is photographed by an encircling or moving camera.

Interlude
A brief, intervening film scene or sequence, not specifically tied to the plot, that appears within a film.

Prologue
A speech, preface, introduction, or brief scene preceding the main action or plot of a film; contrast to epilogue.

Top
404 ERROR (Page Not Found)

A broken path:
If the location of the linked page changes, or if it’s name is changed, the link must be updated to reflect the new path, otherwise a 404 (Page Not Found) error will result.
CSS
Cascading Style Sheets

• Cascading Style Sheets (CSS) is a set of formatting rules that determine the look and position of HTML elements on a page.

• CSS rules are applied to HTML elements based on selectors and declarations.
Selectors

- Selectors determine which HTML element a style should be applied to, such as:

  - tags:<p>

  - classes: can be applied to any element, such as <span class="boldRed">. In the CSS, class names begin with a dot, such as .boldRed.

  - IDs: specify a style for a single, unique element, such as <div id="MainContent">. In the CSS, class names begin with a #, such as #MainContent.

  - pseudo-classes: add custom looks and behaviors to certain selectors, such as a property when the mouse is hovered over a link, or the style of a visited link.
Declarations

- Declarations consist of a property and a value. The property is the style attribute (like color or margin) that you wish to change. Each property has a value (like "red" or "20px").
The CSS Box Model
The CSS Box Model
The CSS Box Model

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent.
The CSS Box Model

• **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent

• **Border** - A border that goes around the padding and content. The border is inherited from the color property of the box, or it can have its own color.
The CSS Box Model

• **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent

• **Border** - A border that goes around the padding and content. The border is inherited from the color property of the box, or it can have its own color.

• **Padding** - Clears an area around the content. The padding is affected by the background-color or background-image of the box
The CSS Box Model

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent.

- **Border** - A border that goes around the padding and content. The border is inherited from the color property of the box, or it can have its own color.

- **Padding** - Clears an area around the content. The padding is affected by the background-color or background-image of the box.

- **Content** - The content of the box, where text and embedded images appear.
The CSS Box Model

The CSS Box Model

box-sizing: content-box;

The CSS Box Model

box-sizing: content-box;

This is the default behavior (no CSS rule needed).

The CSS Box Model

box-sizing: content-box;

This is the default behavior (no CSS rule needed).

Width and height is applied only to the element's content box. Any border or padding is added to the width and height when rendering the object on screen.

The CSS Box Model

box-sizing: content-box;
The CSS Box Model

```css
box-sizing: content-box;

What is the total width of this element?

width: 250px;
padding: 10px;
border: 5px solid gray;
margin: 20px 10px 20px 10px;
```
The CSS Box Model

box-sizing: content-box;

What is the total width of this element?

width: 250px;
padding: 10px;
border: 5px solid gray;
margin: 20px 10px 20px 10px;

250+(10+10)+(5+5)+(10+10) = 300px
The CSS Box Model

The CSS Box Model

box-sizing: border-box;

The CSS Box Model

box-sizing: border-box;

border-box tells the browser to account for any border and padding in width and height. If you set an element’s width to 100 pixels, that 100 pixels will include any border or padding you added, and the content box will shrink to absorb that extra width. This typically makes it much easier to size elements.

The CSS Box Model

box-sizing: border-box;
The CSS Box Model

box-sizing: border-box;

What is the total width of this element?

width: 250px;
padding: 10px;
border: 5px solid gray;
margin: 20px 10px 20px 10px;
The CSS Box Model

```css
box-sizing: border-box;
```

What is the total width of this element?

```css
width: 250px;
padding: 10px;
border: 5px solid gray;
margin: 20px 10px 20px 10px;
```

```
250+(10+10)+(5+5)+(10+10) = 250px
```
The CSS Box Model

box-sizing: border-box;
The CSS Box Model

```css
box-sizing: border-box;

Use this in your universal selector for easier element sizing:

```
Floats and Clears

• Floats cause block elements to behave as inline elements, that is, they will move right or left rather than stacking in their own lines.

• Clears force a line break. In other words, they will cause an inline-behaving object to appear on a new line.

• Objects must have defined widths to float next to each other, or else they will overlap.
Responsive Design

• RWD acknowledges that websites are increasingly accessed from a multitude of devices with different screen sizes and in different contexts - sitting at a desk or on a train, swiping from page to page or having the words read aloud.

• When executed properly, a single responsive site provides optimized users experience and content across all devices.
building flexible sites

1. sketch

2. compute grid and dimensions

3. convert static grid to fluid, pixel measurements to percentages
   • target ÷ context = result

4. build out wireframe

5. discover breakpoints and add media queries
target ÷ context = result

Take the target size from the comp, and divide it by the size of its containing element—in other words, its context. The result is the desired size expressed in relative percentages.
Flexible Grids

• Every row and column can be expressed as a proportion of their containing elements.
Flexible Font Sizes

• Text Default Size: Most browsers interpret 100% as 16px. If the target size of your `<p>` text, for example, is 11px, divide 11px by the font-size of its containing element: $11 \div 16 = 0.6875$em.
flexible type

```
body {
  background-color: #DCDBD9;
  color: #2C2C2C;
  font: normal 100% Georgia, serif;
}
```

not flexible type

```
body {
  font-size: 24px;
  font-style: italic;
  font-weight: normal;
}
```
target ÷ context = result

take the target font size from the comp, and divide it by the font-size of its containing element—in other words, its context. The result is our desired font-size expressed in relative, flexible ems.

base font-size: 100% usually equates to 16px

24 ÷ 16 = 1.5em
flexible type

body {
  background-color: #DCDBD9;
  color: #2C2C2C;
  font: normal 100% Georgia, serif;
}

flexible type

body {
  font-size: 1.5em;
  font-weight: normal;
}
target ÷ context = result

The context of an element is expressed by the parent it is contained within. ie the context of an `<a>` element within an `<h1>` element is the `<h1>` element.

If the desired target for an `<a>` within a 24px (1.5em) `<h1>` is 11px, the formula would be:

\[ 11 ÷ 24 = 0.458333333333 \]

```css
h1 a {  
  font: bold 0.45833em Arial, sans-serif; /* 11px / 24px */  
  color: #747474;  
  letter-spacing: 0.15em;  
  text-transform: uppercase;  
  text-decoration: none;  
}
```
Media Queries

- Media queries inspect the physical attributes on the device being used.

- HTML meta tag - viewport (must be present)
Media Queries

In CSS

```css
@media screen and (min-width: 1024px) {
  body {
    font-size: 100%;
  }
}
```
Media Queries - Break Points

@media screen and (min-width: 1024px)

- At each breakpoint, media queries cause new CSS to be loaded, changing the size and position of elements.

- Test for appropriate break points by enlarging your window until the design no longer works well. Note the window size, create a break point, and alter the CSS as needed.
Misc. Final Steps

• Embed images as `<div>` background-image (in CSS) and in `<img>` elements (in CSS)

• Import web fonts, as needed

• Add CSS decorations as needed: box shadows, borders, rounder corners, color gradients

• Populate content areas with text and images
Image Optimization

• Determine the largest size needed for any given image

• Use the Image Size command to resample an image down to the largest necessary dimension

• Use the Save for Web command to select the most appropriate format (jpg, png, gif) and the most compression finding the balance of image size and quality.
Reference

• HTML5 Reference from MDN

• CSS Properties from MDN
HTML & CSS tools

- caniuse.com
  - browser compatibility
  - some properties require vendor-specific prefixes
  - many designers no longer supporting IE 8 and older
- html5please.com
CSS tools

- css3gen.com
- box shadow
- text shadow
- border-radius.com
CSS tools

- [google.com/fonts](https://google.com/fonts)
- include link in HTML
- Include font-family code in CSS
CSS tools

- Xcode - iOS Simulator
- Google Chrome Dev Tools
- Opera Mobile Emulator
Phase 4: deploy

1. Content Migration
2. User Testing
3. Training
4. Launch