1. Computer Memory: Storing 1’s and 0’s

How do computers store 1’s and 0’s in a system that allows data to be stored, read, and changed? The textbook talks about the various generations of computers. The changes from one generation to the next refer to how bits are stored, read, and changed.

How can a computer store a 9x9 bitmap of the letter T? In binary, this bitmap is 111010010 without its header or color information. How can a computer store this sequence of nine 1’s and 0’s? A computer needs to use some physical quantity that can be in one of two states. Here are some possibilities:

- Magnetism: Magnets have a north pole and a south pole. A sequence of magnets can be used to represent 1’s and 0’s. North up means 1. Hard disks use magnetism. Core memory (see below) uses magnetism.
- Lights: Lights in house windows can be one or off. A dark window can mean 1, light can be 0. Early computers used spots on a TV screen to store 1’s and 0’s.
- Buckets of water/capacitors of charge: A bucket can hold water. A capacitor can hold charge. A full bucket can be 1, empty 0. The memory in most computers these days store charge in tiny tiny capacitors to represent 1’s. Transistors are the faucets that fill and empty these capacitors.
- Current: A wire can be carrying a current or not. A wire with current can be 1, a wire with no current can be 0. The little switches to turn current on and off used to be electro-mechanical relays, then vacuum tubes, and now transistors.

2. Three Ways to Store Bits: Bumps, Magnetism, Electric Charge

Modern computers store data on CDs, on hard disks, and in RAM (computer memory). Each of these media use a different physical system to store the 1’s and 0’s.

**Compact Discs**

A compact disc is a plastic disk with little pits stamped into its surface. The surface is then covered with a thin reflective coating of metal. The bits are read by a laser beam. The laser is aimed at the disk. The flat parts of the surface reflect light, the bumpy places disperse the light. A light receptor responds to laser light. Therefore flats and pits can be used to represent two states, and a laser and a photo-receptor is used to read the surface.

**Hard Disks**

A hard disk is a stack of metal disks, each one covered with a magnetizable surface. 1’s and 0’s are recorded as magnetized regions with the north pole up or the north pole down. A small coil of wire held on a little stick floats on a cushion of air just over the rapidly spinning disk. As the little magnets pass under the coil, the changing magnetic field induces electric currents in the coil. The computer detects those currents and interprets them as 1’s and 0’s. The computer can send current into the coil to create magnetism that is recorded on the surface.
You could store 1’s and 0’s in a row of sinks, like those in public restrooms. To store a 1, you fill the sink, to store a 0, you leave the sink empty. To convert a 1 to a 0, you open the drain and let the water run out. To convert a 0 to a 1, you close the drain and turn on the faucet.

How do you tell if a sink is full or empty? That’s easy, you open the drain and see if water flows out. If so, there was a 1 there, but now the sink is empty, so you have to fill it again.

That’s how modern memory chips work. Each bit has a capacitor and a transistor. The capacitor holds charge, just as a sink holds water. A transistor is a valve that allows charge to flow or not. Combine a transistor and capacitor and you can store one bit.

---

**Animating Web Pages**

**JavaScript 1**

**1. HTML Meets Scratch**

In the first part of the course we learned about programming. We learned about sequences of well-defined steps expressed in a programming language. We had actions, loops, conditionals, boolean expressions, and variables.

*Elements and Attributes*

Then we learned about HTML. With HTML we could tell a web browser what to put on a page and what those things should look like. Each of the things on the page is called an element or an object and the things that specify how these things look are called the properties or attributes of the objects.

But an HTML page is fixed; it does not change. On the other hand, the elements/objects/sprites on a Scratch stage are not fixed. With a script, we can change attributes of sprites. We can write a script to change the costume, color, size, position of a sprite. Can we bring elements on an HTML page to life the way we can bring the sprites on a Scratch screen to life?

Absolutely! All we need to do is write scripts that tell the browser how to change the attributes of the elements on the page. The language browsers use is called Javascript. This language is *not* the same as a language called Java. Please do not confuse them. Javascript is not Java.

*Events and Functions*

In Scratch, we write chunks of code and we tied those chunks of code to events. For example, we could tell scratch to activate a script when the user clicks on the flag, or presses the ’6’ key, or clicks somewhere. Any action that can activate a script is called an event.

In Javascript, we also activate code in response to events.
2. A First Example

This short HTML page shows the essential skills to activating a webpage:

```html
<html>
<!-- slideshow0b.html -->
<head>
<title>A Picture</title>
<script type='text/javascript'>
// this function shows 2.jpg in the img tag with id = pic
function show2()
{
    document.getElementById('pic').src = 'small/2.jpg';
}
</script>
</head>
<body>
<div style='text-align: center'>
A Slide Show
<br />
<a href='big/1.jpg'>
<img src='small/1.jpg' id='pic'/>
</a>
<br />
<iframe height='50' width='300'
    style='border: 0,'
    src='captions/1.html'></iframe>
<br />
<input type='button' value='see slide 2'
onclick='show2()'/>
</div>
</body>
</html>
```

There are three important skills in this page. Most of the page looks like regular HTML, but three parts are new. Starting at the bottom, and working upwards, we see:

- **button** The first new thing is an element of type `<button>` defined at the bottom of the page. This tag puts a button on the screen. The message "see slide 2" appears on the button. When the user clicks the button, we tell the browser to run some code called `show2()` which we discuss shortly.

- **id='pic’** The image tag (`<img>`) has a new attribute: `id='pic'` which gives a name to the image tag. Giving an identifying name to an element allows us to refer to that element in the script. This img tag is the element called `pic` in this HTML document.

- **<script>** Finally, in the head of the page, we put the code. We enclose the code in a container started with `<script>` and ended with `</script>` as shown in the example. Inside of this container is a function called `show2()` which changes the src attribute of the element with id of pic. The notation is similar to the directory paths in Unix except we use dots here instead of slashes.

The document in the top level container, then we specify the element with id of pic, then we specify the src attribute of that element.
Summary

To make a webpage more active, we need to:

1. Assign an id to each element we want to modify
2. Write code in the head of the page to modify the attributes
3. Add attributes to connect events to code

That’s it. All we need to do is get familiar with:

1. The events that can activate code,
2. The names of attributes we can modify
3. How to use variables, loops, conditionals, booleans in Javascript

List of Events and Some Properties

<table>
<thead>
<tr>
<th>Attribute</th>
<th>The event occurs when...</th>
</tr>
</thead>
<tbody>
<tr>
<td>onabort</td>
<td>Loading of an image is interrupted</td>
</tr>
<tr>
<td>onblur</td>
<td>An element loses focus</td>
</tr>
<tr>
<td>onchange</td>
<td>The content of a field changes</td>
</tr>
<tr>
<td>onclick</td>
<td>Mouse clicks an object</td>
</tr>
<tr>
<td>ondblclick</td>
<td>Mouse double-clicks an object</td>
</tr>
<tr>
<td>onerror</td>
<td>An error occurs when loading a document or an image</td>
</tr>
<tr>
<td>onfocus</td>
<td>An element gets focus</td>
</tr>
<tr>
<td>onkeydown</td>
<td>A keyboard key is pressed</td>
</tr>
<tr>
<td>onkeypress</td>
<td>A keyboard key is pressed or held down</td>
</tr>
<tr>
<td>onkeyup</td>
<td>A keyboard key is released</td>
</tr>
<tr>
<td>onload</td>
<td>A page or an image is finished loading</td>
</tr>
<tr>
<td>onmousedown</td>
<td>A mouse button is pressed</td>
</tr>
<tr>
<td>onmousemove</td>
<td>The mouse is moved</td>
</tr>
<tr>
<td>onmouseout</td>
<td>The mouse is moved off an element</td>
</tr>
<tr>
<td>onmouseover</td>
<td>The mouse is moved over an element</td>
</tr>
<tr>
<td>onmouseup</td>
<td>A mouse button is released</td>
</tr>
<tr>
<td>onreset</td>
<td>The reset button is clicked</td>
</tr>
<tr>
<td>onresize</td>
<td>A window or frame is resized</td>
</tr>
<tr>
<td>onselect</td>
<td>Text is selected</td>
</tr>
<tr>
<td>onsubmit</td>
<td>The submit button is clicked</td>
</tr>
<tr>
<td>onunload</td>
<td>The user exits the page</td>
</tr>
</tbody>
</table>

Image Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>Sets or returns how to align an image according to the surrounding text</td>
</tr>
<tr>
<td>alt</td>
<td>Sets or returns an alternate text to be displayed</td>
</tr>
<tr>
<td>border</td>
<td>Sets or returns the border around an image</td>
</tr>
<tr>
<td>complete</td>
<td>Returns whether or not the browser has finished loading the image</td>
</tr>
<tr>
<td>height</td>
<td>Sets or returns the height of an image</td>
</tr>
<tr>
<td>hspace</td>
<td>Sets or returns the white space on the left and right side of the image</td>
</tr>
<tr>
<td>id</td>
<td>Sets or returns the id of the image</td>
</tr>
<tr>
<td>src</td>
<td>Sets or returns the URL of an image</td>
</tr>
<tr>
<td>vspace</td>
<td>Sets or returns the white space on the top and bottom of the image</td>
</tr>
<tr>
<td>width</td>
<td>Sets or returns the width of an image</td>
</tr>
</tbody>
</table>

Iframe Object Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>align</td>
<td>Sets or returns how to align an iframe according to the surrounding text</td>
</tr>
<tr>
<td>height</td>
<td>Sets or returns the height of an iframe</td>
</tr>
<tr>
<td>id</td>
<td>Sets or returns the id of an iframe</td>
</tr>
<tr>
<td>marginHeight</td>
<td>Sets or returns the top and bottom margins of an iframe</td>
</tr>
<tr>
<td>marginWidth</td>
<td>Sets or returns the left and right margins of an iframe</td>
</tr>
<tr>
<td>src</td>
<td>Sets or returns the URL of the document that should be loaded into an iframe</td>
</tr>
<tr>
<td>width</td>
<td>Sets or returns the width of an iframe</td>
</tr>
</tbody>
</table>
Style Attributes

Almost all elements can have a style attribute. The style attribute consists of many many properties. These have names we use in the style attribute, but in Javascript, the names cannot have dashes in them, so there are slightly different spellings of all the same things.

For example, to set the background color of an element, one can write:

```javascript
document.getElementById('thing').style.backgroundColor = 'blue';
```

Here is a pretty complete list of things that can go after the word style.

```
# Color and Background
  * background (CSS)
  * backgroundAttachment
  * backgroundColor
  * backgroundImage
  * backgroundPosition
  * backgroundPositionX
  * backgroundPositionY
  * backgroundRepeat
  * color (CSS)

# Border and Layout
  * border (CSS)
  * borderBottom
  * borderBottomColor
  * borderBottomStyle
  * borderBottomWidth

# Positioning
  * clip
  * display
  * height (CSS)
  * left (CSS)
  * maxHeight
  * maxWidth
  * minHeight
  * minWidth
  * overflow
  * overflowX
  * overflowY
  * pixelBottom
  * pixelHeight
  * pixelLeft
  * pixelRight

# Font and Text
  * font
  * fontFamily
  * fontSize
  * fontStyle
  * fontWeight
  * fontWeight (currentStyle)
  * letterSpacing
  * textAlign
  * text-align
  * text-align-last
  * textAutospace
  * textDecoration
  * textDecorationBlink
  * textDecorationLineThrough
  * textDecorationNone
  * textDecorationOverline
  * textDecorationUnderline
  * textIndent
  * text-justification
  * text-transform
  * text-underline-position
  * verticalAlign
  * white-space
```

Input Elements

Two very useful input elements are the button and the text input element. These are defined with these tags:

```html
<input type='button' value='what it says' style='...'/>

<input type='text' value='what it says' size='width in chars' style='...'/>
```