Overview of Code.org CS Principles

Vision • Goals • Timelines

July 15, 2014

CSTA Annual Conference, St. Charles, IL
Vision • Goals • Timelines

Overview of Code.org CSP
Curriculum Writing Team

- Pat Phillips – editor
  - Code.org

- Rebecca Dovi - writer
  - CodeVA

- Baker Franke - writer
  - University of Chicago Lab Schools

- Andy Kuemmel - writer
  - Madison West High School
Code.org CSP Curriculum

- Why
- What
- Timeline
- Preview
Andy Kuemmelmel

Madison West High School

Madison, WI
Why Code.org is writing a CSP curriculum
Why write another CSP Curriculum?

- Why does anyone write a curriculum?
- Who are we writing it for?
- Facing rapid expansion of requests for CS Courses and curricula
What we’re creating
What is it?

- Head to tail curriculum
- Teaching Guide/Manual
  - Modeled on ethos of ECS
  - Equity • Inquiry • CS Content
- Code.org online tools that blends with curriculum
- Programming: javascript / web applications
  - Both block and text based
- Preparation for PTs
- PD & Community of Practice
# Head to Tail Curriculum

## Unit 1 - Sending Bits

The first unit begins with the question, how do we get a single bit of information from one place to another? and explores the technical challenges and questions that arise from the need to transfer information between people and computational devices.

**Start:** Students invent a method of sending a single bit of data.

**End:** Students invent their own binary encoded, compressed file type to represent some information of their choosing.

**Assessment:** Bits, bytes, abstract data

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bits, bytes, abstract data</td>
</tr>
<tr>
<td>2</td>
<td>Bits, bytes, abstract data</td>
</tr>
</tbody>
</table>

## Unit 2 - Algorithms

The algorithms unit builds students’ problem solving skills through unplugged and computer-based problems. Students are asked to express algorithmic solutions with increasing formality of language and to analyze algorithms.

**Start:** Unplugged unplugged development, exploration, and analysis of classic algorithmic problems e.g. sorting.

**End:** Block-programming implementation of asymmetric algorithms e.g. public key cryptography

**Assessment:** Algorithm analysis & design

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Unplugged unplugged development, exploration, and analysis of classic algorithmic problems e.g. sorting.</td>
</tr>
<tr>
<td>4</td>
<td>Block-programming implementation of asymmetric algorithms e.g. public key cryptography</td>
</tr>
</tbody>
</table>

## Unit 3 - Programming

The programming builds on algorithmic thinking and problem solving skills and introduces a formal programming language, JavaScript. Programming sequence enables students to write simple programs to be executed in web pages.

**Start:** Creation of programs in a controlled/sandbox environment that solve problems.

**End:** Complete a practice data performance task using the programming constructs learned in the unit.

**Practice PT:** Programming

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Creation of programs in a controlled/sandbox environment that solve problems.</td>
</tr>
<tr>
<td>6</td>
<td>Complete a practice data performance task using the programming constructs learned in the unit.</td>
</tr>
</tbody>
</table>

## Unit 4 - Data

The data builds on students’ understanding of structured data begun in the previous unit. Students will develop an understanding of the nature and structure of data that enables them to ask and answer questions of the data effectively and responsibly. An exemplary data investigation will help prepare students for the PT.

**Start:** How to find and understand real data.

**End:** Complete the investigate performance task.

**Investigate PT:** (data)

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>How to find and understand real data.</td>
</tr>
<tr>
<td>8</td>
<td>Complete the investigate performance task.</td>
</tr>
</tbody>
</table>

## Unit 5 - Innovation

The innovation unit synthesizes the topics covered in the previous units and leads students through a practice and then the formal performance task of researching and understanding the technology behind some modern innovations.

**Start:** Practice and skill building with researching relevant innovations.

**End:** Complete the create performance task.

**Explore PT:** (innovation)

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>Practice and skill building with researching relevant innovations.</td>
</tr>
<tr>
<td>10</td>
<td>Complete the create performance task.</td>
</tr>
</tbody>
</table>

## Unit 6 - Web Apps

The web apps unit builds on previous work with programming to build creative web apps in an open-ended IDE that supports both text and visual programming. Students will learn about the structure of modern web applications and build a few small examples of certain types of applications.

**Start:** Transition from block-based algorithm creation to writing web applications in an IDE that supports block- and text-based programming.

**End:** Complete the create performance task.

**Create PT:** (programming)

<table>
<thead>
<tr>
<th>Day</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Transition from block-based algorithm creation to writing web applications in an IDE that supports block- and text-based programming.</td>
</tr>
<tr>
<td>12</td>
<td>Complete the create performance task.</td>
</tr>
</tbody>
</table>

---

**Key Concepts:**

1. Creativity
2. Abstraction
3. Data
4. Algorithms
5. Programming
6. Internet
7. Impact
Connecting Theme: Internet & Innovation

Unit 1: Sending Bits
The first unit begins with the question: how do we get a single bit of information from one place to another? It explores the technical challenges and questions that arise from the need to transfer information between people and computational devices.
Start: students invent a method of send a single bit of data
End: students invent their own binary encoded, compressed file type to represent some information of their choosing
Assessment: Bits, Bytes, Abstract data
20 days

Unit 2: Algorithms
The algorithms unit builds students' problem-solving skills through unplugged and computer-based problems. Students are asked to express algorithmic solutions with increasing formality of language and to analyze algorithms.
Start: unplugged/plugged development, exploration, analysis of classic algorithmic problems e.g. sorting
End: block-programming implementation of asymmetric algorithms e.g. public key cryptography
Assessment: Algorithm analysis & design
25 days

Unit 3: Programming
The programming builds on algorithmic thinking and problem-solving skills and introduces a formal programming language, JavaScript. Programming sequence enables students to write simple programs to be executed in web pages.
Start: creation of programs in a controlled/sandbox environment that solve problems
End: complete a practice data performance task using the programming constructs learned in the unit.
Practice PT: Programming
30 days

Unit 4: Data
The data builds on students' understanding of structured data from the previous unit. Students will develop an understanding of the nature and structure of data that enables them to ask and answer questions of the data effectively and responsibly. An exemplary data investigation will help prepare students for the PT.
Start: how to find and understand real data
End: complete the investigate performance task.
Investigate PT (data)
30 days

Unit 5: Innovation
The innovation unit synthesizes the topics covered in the previous units and leads students through a practice and then the formal performance task of researching and understanding the technology behind some modern innovations.
Start: practice and skill building with researching relevant innovations
End: complete the explore performance task.
Explore PT (innovation)
15 days

Unit 6: Web Apps
The web apps unit builds on previous work with programming to build creative web apps in an open-ended IDE that supports both text and visual programming. Students will learn about the structure of modern web applications and build a few small examples of certain types of applications.
Start: transition from block-based algorithm creation to writing web applications in an IDE that supports block- and text-based programming
End: complete the Create performance task.
Create PT (programming)
25 days
Teaching Guide/Manual

- Daily lesson plans
- ~150 days of school
- Mapped to CSP LOs and EKs
  - Mapped to other standards
- Modeled on ethos of ECS
  - Equity • Inquiry • CS Content
- Variety of instructional strategies
Blended Instruction
Blended Instruction

- Interactive, game-like instructional widgets
- Think: CS Unplugged Activity, plugged in
- An “eco-system” of tools aid instruction and allow students to play and practice with concepts
- Student activity increasingly programming-like, but situated in concrete, problem-solving contexts related to CSP topics
Programming

- Javascript
- Blockly / text transition
- Goals:
  - Integrated into data tasks
  - Computational artifacts are web applications
  - Focus on skills needed to produce good artifacts for PTs
Videos

- 8 videos planned tied to CSP curriculum
- [filming starts in August]
- Videos:
  - Explain concepts that are hard to experience
  - Try to tell a story
  - Be used in curriculum to reinforce and extend rather than as primary instruction
  - Have some kind of ‘celebrity’ factor
Preparation for Performance Tasks

- PT-like sequences and tasks built into the curriculum
- Model performance tasks
- Students should be prepared to tackle PTs on their own
- Focus on writing, communication and collaboration skills
Professional Development & Community

- PD modeled on ECS
  - 4 Phases over 15 months
  - In-person PD cohorts tethered to facilitator experts
  - Ongoing, Job-embedded PD

- **TO DO:**
  - Integrated teaching guide support / feedback
  - Curriculum-embedded cohort support & engagement
A Word on Evaluation

- **All** Code.org Curriculum and PD is evaluated by Outlier research

- Implementation measurement
- Evaluation of efficacy
- Measure Spread and Sustainability of Innovation
A Taste of AP CSP
Rebecca Dovi

CodeVA

Richmond, VA
Abstraction

2.2.3F -- A logic gate is a hardware abstraction that is modeled by a Boolean function.

2.2.3I -- Hardware is built using multiple levels of abstractions, such as transistors, logic gates, chips, memory, motherboards, special purposes cards, and storage devices.
Squishy

Why doesn't it light up?
Baker Franke

Code.org

Chicago, IL
Text Compression

Choose a poem: Pitter patter pitter...

Compressed:
Pitter_patter_pitter_patter_liste_to_the_rain_pitter_patter_pitter_patter_on_the_window-pane

poem size: 93 bytes
dictionary size: 0 bytes
total: 93 bytes
compression: 0%

NAME:

Dictionary:

Go To Widget
Text Compression

- Big Ideas Covered?
Baker’s Big 3 Keys to CSP Curriculum

- Abstraction is the key that unlocks computational thinking
- Programming to the level of “heavy scripting”
- Writing ability correlates with success on PT [sneaky]
Unit 5: Innovations
Timelines
"This is our plan for the next 1,000 years."
In development:

- Plan summer PD institute for pilot teachers
- Plan academic-year support for pilot teachers
- Plan follow-up summer PD institute for pilot teachers
- Writers begin to revise lessons and assessments from units 1-3 based on piloter feedback
- Writers begin to revise lessons and assessments from units 4-6 based on pilot feedback

Available for use:

- Spring 2014: Subset of units 1-3 available online
- Summer 2014: Summer PD institute for pilot teachers
- Fall 2014: Start of academic-year support for pilot teachers
- Winter 2014/15: Revised copy of units 1-3 available online
- Spring 2015: Revised copy of units 4-6 available online
- Summer 2015: Follow-up summer PD institute for pilot teachers
Timeline

- Summer 2014
  - finish draft, begin pilot PDs

- Fall 2014
  - piloting / development / beta release of curriculum in sequence

- Winter/Spring 2015
  - hopefully tool development reaches early phases of completion

- Summer 2015
  - roll out PD in partner districts with capacity
  - release version 1 of curriculum publicly
Thank you

Q&A